# Robot Communication: Issues and Implementations

by

Holly A. Yanco

B.A. Wellesley College (1991)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Science in Computer Science and Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 1994



Author .....
Department of Electrical Engineering and Computer Science
May 12, 1994

Certified by .....
Lynn Andrea Stein
Class of 1957 Assistant Professor of Computer Science
Thesis Supervisor

Accepted by .....
Frederic R. Morgenthaler
Chairman, Departmental Committee on Graduate Students

# Robot Communication:

# Issues and Implementations

by

Holly A. Yanco

Submitted to the Department of Electrical Engineering and Computer Science
on May 12, 1994, in partial fulfillment of the
requirements for the degree of
Master of Science in Computer Science and Engineering

## Abstract

In this thesis, we demonstrate that robots can develop their own languages for communication. We call the languages developed by the robots Adaptable Synthetic Robot Languages (ASRLs). The basic ASRL is a simple one-to-one mapping of signals to concepts. We then present two ASRL types that use small amounts of additional structure to reduce learning times for ASRLs. In the context dependent ASRL, the robots use sensor values to help determine the meaning of a signal. In the compositional ASRL, the robot uses a simple grammar to build up concepts from words.

The robots develop their languages using reinforcement learning. Since the language learning task is a dependent problem, the traditional method of individual reinforcement fails. We introduce task-based reinforcement, which is able to handle dependent learning problems.

Thesis Supervisor: Lynn Andrea Stein
Title: Class of 1957 Assistant Professor of Computer Science

# Acknowledgments

First and foremost, I would like to thank Lynn Stein for her guidance during the past three years.

To my parents, I present my "paper." Mom and Dad, it's finally finished!

Thanks to Nancy Ross, who taught me that robots need crêpes.

Many people have been supportive throughout the course of this thesis. They have listened to practice talks, read drafts of the thesis, given useful advice, and provided distractions. These people include Dave "Bud" Baggett, David Beymer, Rod Brooks, Joanna Bryson, Matt Domsch, Greg Galperin, Andy Gavin, Eric Grimson, Ian Horswill, Tina Kapur, Greg Klanderman, Lynne Parker, Nancy Pollard, Joe Marks, Maja Mataric, Brian Reistad, Jason Rubin, Marika Santagata, Tim Tang, Peter Todd, Mark Torrance, Pearl Tsai, Mike Wessler, Stewart Wilson, Rick Yanco, Catherine Young, the members of How to Solve AI, and everyone that I have inadvertently omitted from this list.

Special thanks to Oded Maron.

The author was partially supported by Digital Equipment Corporation. This material is based upon work supported by the National Science Foundation under National Science Foundation Young Investigator Award Grant No. IRI–9357761. Any opinions, findings, conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation. Professor Stein's research is also supported by the Gordon S. Brown fund, by the Mitsubishi Electric Research Laboratories, by the General Electric Foundation Faculty for the Future Award, and by the Class of 1957 Career Development Chair.

The research described here was conducted at the Artificial Intelligence Laboratory of the Massachusetts Institute of Technology. Support for the laboratory's artificial intelligence research is provided in part by the Advanced Research Projects Agency of the Department of Defense under Office of Naval Research contract N00014–91–J–4038.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

This thesis gives evidence that robots can develop languages for communication. We introduce a new reinforcement technique better suited to multi-agent tasks involving dependent learning problems. Using this technique, we demonstrate that both physical and simulated robots can learn adaptable synthetic robot languages (ASRLs). We show further the introduction of small amounts of additional structure – context and grammar – significantly decreases the amount of time required to learn ASRLs.

## 1.1 Overview of the problem

Robots can and will fail.

We can use multiple robots instead of relying on one all-purpose robot. The group of robots may work together to complete a task more expediently or simply act as backups for one another. Most often, the robots work as a team to reach their goals.

However, when we choose to use multiple robots instead of a single robot, we have a new problem to consider: communication. When multiple robots must cooperate in order to complete a task, communication may expedite cooperation. Some tasks may not even be able to be completed without communication. For example, if only one of the robots knows which task needs to be completed and can not complete the task by itself, this robot needs to find a way to communicate this information to the other robots in order for the task to be accomplished.

In previous work with multiple agents, robots are usually given a language developed by a human programmer. However, the provided language may not be well-suited to the robots or to the tasks they are to perform. A programmer may not anticipate all of the communication needs of the robots; if the robots encounter novel situations they can not talk about, the mission could fail. Robots that develop their own languages could overcome this problem since they develop their language as they try to complete tasks in the world. Robots usually can not adapt human provided languages. Robot developed languages will be adaptable since once the robots have the ability to develop a language, they have the ability to modify that language.

In order to develop a language, the robots need to be able to learn. The development of language can be viewed as a learning problem in which a group of robots learn to agree on an interpretation for signals. We call the communication languages that our robots develop Adaptable Synthetic Robot Languages (ASRLs). We call them "adaptable" because the languages can change in dynamic environments and "synthetic" to distinguish them from so-called natural languages (like English) that people use. We will present three types of ASRLs that the robots can develop: *basic*, *context dependent*, and *compositional*. The context dependent and compositional ASRLs are extensions of the basic ASRL in which we use additional structure to speed up learning.

We have chosen to use reinforcement learning as our learning method since it provides incremental learning. We also selected this method since we were interested in exploring multi-agent reinforcement learning. In reinforcement learning, an agent selects an action to perform on a given input. The agent then receives a reinforcement value which tells the agent if it has acted properly. In our work, the robots receive either a "good robot" or "bad robot" reinforcement. In reinforcement learning, reinforcement values can vary over a spectrum of values and need not be binary. (We discuss reinforcement learning in depth in Chapter 3.)

In single agent reinforcement learning, it is clear that the lone robot deserves all of the credit if a task is completed properly. In multi-agent reinforcement learning, how should the robots be reinforced? One traditional method gives each robot an

individual reinforcement based only on their individual action. It seems that this method would guarantee reliable reinforcement, but we will show that this method fails to work for dependent learning problems. We introduce a new method for reinforcement, which we call *task-based reinforcement.* In this method, the group of robots only receives one reinforcement value. The value is positive only if the robots complete the given task and negative if the robots do not complete the task properly. This method requires only a simple decision to determine reinforcement: was the task completed properly? Conversely, the individual reinforcement method requires someone (or something) to determine each individual robot's contribution towards the completion of the desired task. We will show that our new reinforcement method is effective in the language learning task and that it succeeds where individual reinforcement fails.

## 1.2 Overview of the thesis

In Chapter 2, we describe the experimental scenario, discuss the experimental assumptions, and discuss our methods for data collection.

In Chapter 3, we give an overview of reinforcement learning and discuss the particular algorithm that we have used, Kaelbling's Interval Estimation. We also present two multi-agent reinforcement schemes: individual reinforcement and task-based reinforcement.

The following three chapters present communication experiments in the development of ASRLs and results. Chapter 4 describes the basic ASRL experiments. In Chapters 5 and 6, we show how adding additional structure to the language improves learning times. Chapter 5 discusses the context dependent ASRL experiments where the additional structure involves using sensor values to determine the meanings of signals. Chapter 6 presents the compositional ASRL experiments where the additional structure is a simple grammar. Each chapter also suggests directions for future work on the particular ASRL type.

In Chapter 7, we discuss how the work described in this thesis relates to previous

work in several areas of Artificial Intelligence and Computer Science.

Our conclusions are in Chapter 8.

Appendix A describes the robots we used in a few of the basic ASRL experiments. Appendix B contains data tables for points graphed in the thesis.

# Chapter 2

# Experimental setup

In this chapter, we describe the methods we have used in our experiments. First, we discuss the assumptions that we have made. Then we present our experimental language development scenario and discuss our data collection mechanisms.

## 2.1 Experimental assumptions

Since we have focused on language learning in this thesis, we have made several assumptions to simplify the problem:

**Assumption 1** We assume that communication is perfect. In fact, in our work on real robots, we did have perfect communication within a 10-foot range.

**Assumption 2** We assume turn-taking — i.e. the robots will not talk over one another. Turn-taking is the mechanism that allows communication between two or more people to be heard and understood. If everyone were to talk at once, many of the messages would be lost and many of the messages that got through might be incorrect. This assumption is necessary to ensure perfect communication. If more than one robot could broadcast at the same time on the same channel, the communication would be very noisy.

**Assumption 3** The robots do not acknowledge messages from other robots. Since the communication is assumed to be perfect, we assume that the signal is received by the intended robotic audience.

The issues we ignore by using assumptions 2 and 3 have been and are currently extensively researched in the Distributed Systems area (see, for example, [Mullender, 1993]).

**Assumption 4** We do not address plan recognition in which an agent attempts to determine the goals of another agent by watching the other agent. Plan recognition can be thought of as another form of communication, much as humans rely on body language to offer clues to what another person is thinking or feeling.

**Assumption 5** We assume that the robots have an audience to address. Without this assumption, the robot sending a signal to other robots would not know if there were other robots listening to it unless it was able to recognize its fellow robots (which is known as kin recognition).

**Assumption 6** In our experiments, the tasks the robots are to perform are atomic. Atomic tasks do not require planning and complicated task decomposition. Since we primarily want to investigate the language problem, this atomic action set is useful. In physical robots, the atomic actions we use are movement based. Our work in simulation continued the action paradigm for consistency.

While we have concerned ourselves with the development of the language, other researchers have explored how systems can learn to build up the actions that we assume the robots have at the outset. Just as children seem to learn to interact with the world through actions before they start to talk about the world, a robotic system could be developed that learned how to act in the world and then start to learn to talk about what it has learned. Both [Drescher, 1993] and [Drescher, 1991] address the development of primitives using the Schema mechanism which builds up representations of observations about the world. [Pierce, 1991] and [Pierce and

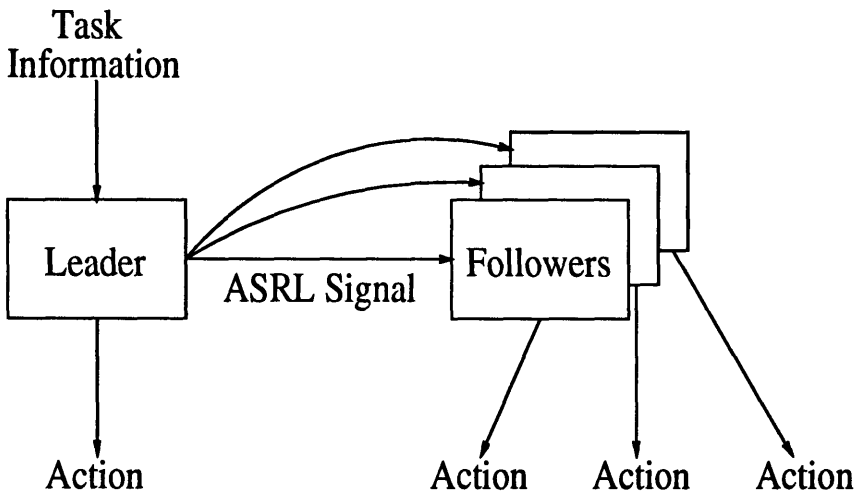



Figure 2-1: Flow of the signals in the experiments

Kuipers, 1991] discuss how robots can learn to use their motors to move around the world.

**Assumption 7** All reinforcement is immediate. Together with Assumption 6, this means that reinforcement will always pertain to the most recent action taken. This assumption simplifies the learning problem and allows us to focus on the development of ASRLs. It could be relaxed through the use of temporal differencing methods such as [Sutton, 1984].

## 2.2 Language development scenario

The language development experiments were performed with the following experimental scenario, illustrated in figure 2-1. We have a group of robots that need to perform a task. The task information is only given to one of the group members; this robot is the *leader*. The leader must communicate the task information to the rest of the group, who are *followers*.

The leader and followers share a common set of action capabilities. The leader also has a fixed but uninterpreted set of signals that can be sent to the followers. The group as a whole needs to agree on mappings from the signals to concepts representing tasks. In these experiments, the concepts map to actions. The leader must also learn

Leader's program:

1. Listen for task information
2. On task signal,
   (a) select signal to send to followers
   (b) send selected signal to followers
   (c) select action
   (d) perform action
3. Wait for reinforcement
4. On reinforcement, update learning tables
5. Goto 1

Follower's program:

1. Listen for robot signal
2. On robot signal,
   (a) choose an action
   (b) perform the selected action
3. Wait for reinforcement
4. On reinforcement, update learning tables
5. Goto 1

Figure 2-2: Description of the programs running on the robots in the experiments.

a mapping from the task information to its action and to the signal it needs to send to the followers.

The programs running on the robots require them to sit and wait until they hear a signal. The leader waits to hear the task signal while the followers wait for an ASRL signal. Once the leader receives the task signal, it selects a signal to send to the followers and sends it. After sending the signal, the leader selects an action to perform and executes that action. When a follower receives an ASRL signal, it selects an action to perform and executes it. After the robots act, each waits to receive a reinforcement value, after which it updates its reinforcement tables. (The control of the robot programs is given in figure 2-2.)

The task information given to the leader may come from a human signal if a human is running the experiments or may come from the environment if the robots are deciding what to do based on the state of the world. An example of a human signal is "move the furniture." An environmental signal may come in the form of a

threat to the robots' safety. It could also come from observations about the world — e.g. "the wastebaskets are overflowing in the offices, so we should clean up."

Let's consider a simple example in the framework of our experimental scenario. We'll have one leader and two followers. The robots need to learn a two concept language; i.e. they need to be able to understand two signals and execute the proper task on those signals. The two tasks will be "all spin" and "all straight". For the "all spin" task, each robot must choose to spin in place. Similarly, for "all straight", each robot must move forward.

In this example, the leader receives a signal from a human which tells the it what task it needs to perform. Initially, the leader does not understand the human signal, so it will need to learn the proper behavior as it tries to act on human commands.

For example, the human gives the leader "all spin" as the first command. The leader will then choose a signal to send to the followers. The leader selects a signal from a set of initially uninterpreted signals that it is given. The leader makes this choice by using previous experiences; if it has acted properly before, it will use this information to try to act properly again. This selection of the "proper behavior" is made using reinforcement learning; we discuss this technique in Chapter 3. The leader also needs to select an action to perform, again using reinforcement learning. As the learning begins, no action or signal is better than any other, so the robot will select randomly from the possible signals and actions.

After the leader has sent a signal, the followers will hear this signal and each needs to act upon it. To do this, each follower needs to select the best action from its action set to perform given the ASRL signal from the leader robot. Again, as the learning begins, each follower will select an action randomly until it begins to receive positive reinforcement informing it that certain actions are better than others.

The robot ASRL is developing as the leader is learning to understand the human input and the followers are learning to understand the leader. This concurrent learning introduces dependencies in the learning that cause simple individualized reinforcement methods to fail. We will discuss this problem in Chapter 3.

There are several things that are being learned in this scenario. The leader needs

to learn to interpret the task information given to it. This interpretation involves performing some action and sending a signal to the followers. The followers need to learn to interpret the leader's signals, which may have differing meanings during the initial evolution of the language. For the followers, interpretation involves only performing an action. The signals that the leader sends to the followers are what we call the ASRL.

In our experimental scenario, only one of the robots can be the leader. An alternate method would allow all of the robots to assume the leadership role. Whenever a robot had information it needed to communicate to the other group members, it would take on the leadership position. Once it had communicated its information, the robot would go back to acting as a follower. We have not implemented this alternate scenario in the current research.

## 2.3 Data collection methods

For data collection, all experiments have been done in simulation. Average times to convergence for all experiments were collected over 100 runs of the experiment.

We define convergence to be the point at which all of the inputs have each been acted upon properly three times. This measure of convergence tends to inflate the number of iterations to convergence. However, since we use this definition consistently, all of the results are comparable to one another.

Once the reinforcement learning has converged upon a solution, it will continue to execute that solution until it receives an amount of negative reinforcement significant enough to outweigh the previous positive reinforcement that has been received. Once all of the inputs have been acted upon properly three times, the learning algorithm will not try new actions for those inputs unless the robot begins to receive negative reinforcement, indicating that it needs to adapt to changing circumstances.

# Chapter 3

# Reinforcement Learning

## 3.1 Introduction

If we want robots to be adaptable, they need to have the ability to learn. Many different learning methods have been developed and are currently being explored[1]. In this work, we have chosen to use reinforcement learning. One advantage of reinforcement learning is that it is an on-line method; i.e. the robots can learn incrementally as they explore, whereas other methods require large batches of examples from which to learn. Another advantage is that the world only needs to provide a reinforcement value that tells the agent how good the action performed was, while in most supervised learning methods, a teacher must tell the agent the correct action.

We can think of reinforcement learning in terms of human experience. If a parent wants to teach a child to make the bed, the parent needs to provide some form of positive reinforcement to the child for a task well done. The parent may also wish to provide positive reinforcement if the child merely makes an attempt at the task. The key is that the child receives feedback from the world, in this case through the parent, that serves as an indicator to whether the child is acting properly. Positive reinforcement is a reward such as an ice cream cone. Negative reward is a penalty such as not being allowed to play Nintendo.

[1] For an overview of machine learning, see [Shavlik and Dietterich, 1990].

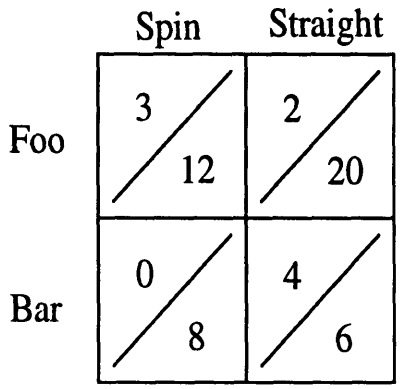



Figure 3-1: Sample reinforcement learning table for two inputs, "Foo" and "Bar", and two outputs, "Spin" and "Straight."

In reinforcement learning, the learning agents try to maximize future rewards based on experience. In other words, once the child learns that making the bed brings good results and not making the bed brings bad results, the child will want to continue to make the bed since that will bring the positive results.

The learner maintains an $input \times output$ table to store information on what outputs have been tried for particular inputs and the reinforcement values that were received. A sample reinforcement table is given in figure 3-1. The numerator keeps track of the amount of reinforcement that has been received and the denominator stores the number of times that output has been tried for the given input. We see that on input "foo", the agent has tried "spin" 12 times and has received positive reinforcement 3 times for this action. In our experiments, reinforcement is either 0 or 1 where 0 is negative reinforcement and 1 is positive reinforcement. Many other reinforcement payoffs are possible. The fractions 3/12 and 2/20 are used to select the action that should be executed the next time input "foo" is seen.

When the robot needs to select an action to perform, it looks for the row in the table that corresponds to the current input. For example, if the current input is "foo", it looks the first row in the table. It uses the values in this table to determine the best action to perform. (While our figure represents the tables as containing fractions, the robots actually keep two tables – one for reinforcement values and one for the number of trials.) The best action may be the highest ratio, the action that has been tried the

least, or some other criterion. The particulars of the action selection depend upon the reinforcement learning technique the agent is using. In this work, we have selected Kaelbling's interval estimation method.

Reinforcement can either be immediate or delayed by some number of time steps. For example, in chess, an agent only receives reinforcement at the end of the game. If the agent wins the game, it receives a positive reinforcement value. If it loses, it receives a negative reinforcement value. At the time the agent receives reinforcement, it does not know which of the moves it made during the game helped it to win. Delayed reinforcement requires the agent to assign credit to particular moves that it made during the course of the game. Temporal differencing (TD) is a reinforcement method which can handle this *credit assignment problem.* For examples of TD work, see [Sutton, 1984], [Sutton, 1988], [Watkins, 1989], [Watkins and Dayan, 1992], and [Tesauro, 1992].

We assumed immediate reinforcement to avoid this problem of temporal credit assignment. The robots are told if they have performed the task properly as soon as they try to perform it. We ensure this immediate reinforcement in our programs by requiring the robots to wait for reinforcement after completing a task before moving on to the next task.

As we described in section 3.3, multi-agent learning presents a credit assignment problem similar to the one described above for delayed reinforcement. However, rather than assigning credit across a series of moves when a delayed reinforcement value is received, a multi-agent learning algorithm must assign credit across a number of agents. Our work in this thesis introduces a reinforcement method, task-based reinforcement, that does not need to assign credit to particular robots.

For some overviews of reinforcement learning, see [Kaelbling, 1993], [Watkins, 1989], [Sutton, 1992], and [Matarić, 1991].

## 3.2 Kaelbling's Interval Estimation

The reinforcement learning method used in this research is Kaelbling's interval estimation [Kaelbling, 1993]. In interval estimation, two tables of *inputs* × *outputs* are maintained. (In the example in figure 3-1, we represented these two tables as one table of fractions.) One of the tables stores the number of times that an output has been performed on a given input and the other stores the number of times that positive reinforcement has been received when performing that output for the input. Each time an input is received, the expected "best" output is selected and the counter for that input/output pair is incremented. If positive reinforcement is received, a second counter for that input/output pair is also incremented. The "best" output given some input is selected by the *ub* optimization function, which is given in figure 3-2. This function favors actions which have received positive reinforcement. However, even if one output results in positive reinforcement, the algorithm will continue to explore the untried outputs in an attempt to attain the maximum reinforcement. When no output has received positive reinforcement yet, outputs are selected randomly.

We extend the algorithm given in figure 3-2 in three dimensions. We allow for multiple inputs by selecting the appropriate line in the table for a given input, where the basic algorithm in figure 3-2 is for only one input. We also allow for more than two outputs by extending the tables in this dimension. To extend this algorithm for multiple agents, each of the agents maintains its own set of tables from which it uses the optimization function to select the best action.

In the interval estimation reinforcement learning method, the agents will continue to explore the space of *inputs* × *actions* until it has experienced everything at least once. Once the agent is convinced that it has found the action that brings the most reward, it will converge upon that action as the best action for the input and begin to exploit this discovery.

During this exploitation phase, agents may begin to receive negative reinforcement for outputs that formerly produced positive results. In order for agents to start exploring again, the amount of negative reinforcement must outweigh the amount of

The initial state, $s_0$, consists of the integer variables $x_0$, $n_0$, $x_1$, and $n_1$, each initialized to 0.

```
u(s,a,r) = if a = 0 then begin
                x0 := x0 + r
                n0 := n0 + 1
           end else begin
                x1 := x1 + r
                n1 := n1 + 1
           end

e(s) =     if ub(x0,n0) > ub(x1,n1) then
                return 0
           else
                return 1
```

where

$$ub(x,n) = \frac{\frac{x}{n} + \frac{z^2_{\alpha/2}}{2n} + \frac{z_{\alpha/2}}{\sqrt{n}}\sqrt{(\frac{x}{n})(1-\frac{x}{n}) + \frac{z^2_{\alpha/2}}{4n}}}{1 + \frac{z^2_{\alpha/2}}{2n}}$$

and $z_{\alpha/2} > 0$.

Figure 3-2: Kaelbling's original interval estimation algorithm from page 52 of [Kaelbling 93]. This algorithm works on one agent that is trying to learn which of two actions should be performed on one input. We extend this algorithm to multiple inputs over multiple agents.

positive reinforcement. However, if the agents have received a large amount of positive reinforcement, they must receive an equivalent amount of negative reinforcement. One solution to this problem is the use of memory windows in learning in which agents only remember a fixed number of previous steps. Another solution would throw out learned values as soon as an unexpected reinforcement value is received; however, this method could have problems in an environment where incorrect reinforcement is occasionally received. We did not implement a relearning strategy in our use of the interval estimation algorithm since we were mostly interested in the initial development of the ASRLs.

## 3.3 Reinforcement of multiple agents

Multi-agent learning is a relatively new subproblem in the reinforcement learning field. It introduces the issue of how to reinforce more than one agent; as discussed above, this is a credit assignment problem. In single agent learning, it is obvious that the single agent should receive the reinforcement. However, in multi-agent learning, which of the agents should receive positive reinforcement? Should it be only the robots that contribute towards the completion of a task? If so, how do we make this determination? Alternatively, should all robots only receive positive reinforcement if the task is completed? Previous work in multi-agent learning is discussed in Chapter 7.

### 3.3.1 Credit assignment in dependent learning problems

We define a learning problem to be *dependent* when the success or failure of one agent's actions depends on the actions undertaken by other agents. The language learning problem addressed in this thesis is dependent since all of the agents need to agree on the ASRL. Learning time for a dependent problem increases as the number of agents increases. During the initial random selection phase, additional agents decrease the chance that the agents will hit upon a good random selection quickly.

In a dependent problem, we need to determine the proper reinforcement to give

the agents. How can we decide which robots should receive positive reinforcement and which should receive negative reinforcement? We'll consider a situation with children and an adult first, and then discuss how that problem relates to this multi-agent learning problem.

A child has several friends over to play. After a few hours, the child's parent comes into the room to find that the floor is covered with toys. Since the parent hopes to avoid having to clean the room, the children are offered ice cream if the room is cleaned. The parent then leaves the children to complete the task. When the parent returns, if the room is cleaned, which of the children should receive ice cream? The task has been completed, so all can receive reward. However, it may be the case that only one child cleaned while the others continued to play. How can the parent know which children contributed towards the goal of a clean room? If the parent only cares whether or not the room is clean, and rewards on the basis of that task being completed, the reinforcement method used is *task-based reinforcement.* However, if the parent offers ice cream only to the children who helped clean, the children will receive *individual reinforcement.* Individual reinforcement requires that the parent remain in the room to determine which of the children are helping to reach the group goal. (We assume that the children will not give accurate information when asked who helped to clean the room since they want to receive ice cream even if they did not help clean.)

The multi-agent learning problem is basically the same as the problem described above. If we give a group of robots a task that needs to be accomplished, we need to determine how to assign reinforcement. This introduces a credit assignment problem analogous to the temporal credit assignment issues of delayed reinforcement. The traditional approach is individual reinforcement. For dependent problems, this breaks. In the next section, we show why it does not work, and in the following section, we introduce task-based reinforcement and show how it solves the credit assignment problem in dependent learning problems.

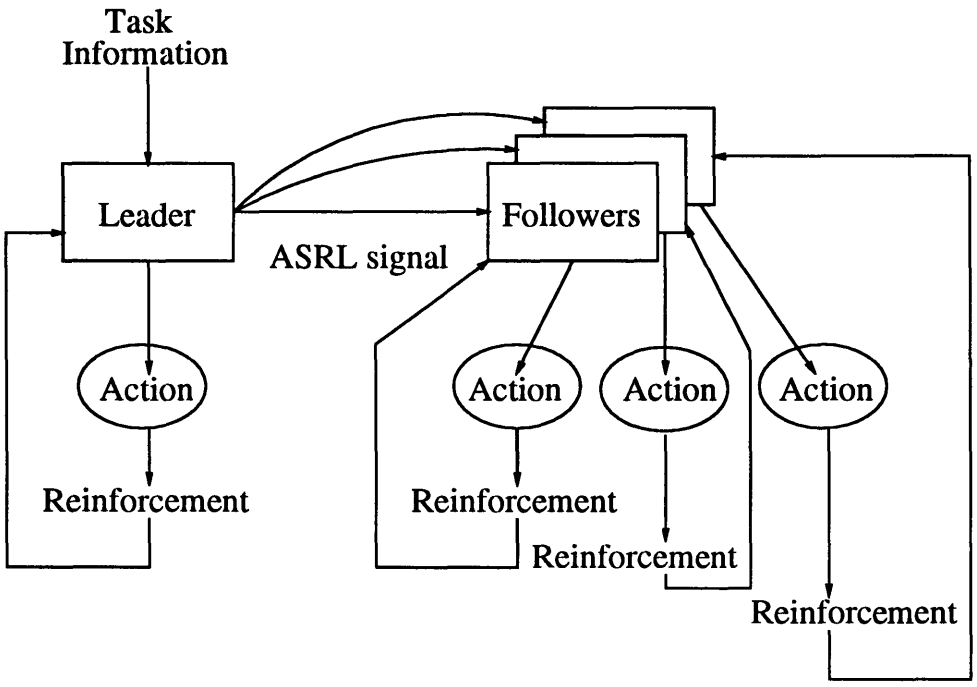



Figure 3-3: Flow of the signals in the experiments using individual reinforcement.

### 3.3.2 Individual reinforcement

Individual reinforcement is a traditional method for providing reinforcement to a group of agents. Each agent receives a reinforcement value that depends only on the actions it has performed. If it is possible to decompose the task into clearly defined subtasks that can be assigned to agents in the group, this method has the advantage of providing an accurate reinforcement value to each robot. However, even if this determination can be made, it may require a large amount of overhead. Additionally, for tasks that are dependent learning problems, it is often difficult to determine which agent contributed towards the group goal. The development of ASRLs is a dependent learning problem.

A diagram of our experimental scenario using individual reinforcement is given in figure 3-3. Note that the individual reinforcements depend only on each agent's action. The leader receives reinforcement for the ASRL signal, but the particular signal that is sent does not matter in the reinforcement value calculation for each individual agent. This causes a problem for individual reinforcement in this experimental scenario.

Let's consider what can happen when two followers receiving individual reinforce-

ment are trying to converge upon a language with the leader of the group. We have three robots, one leader and two followers, that are trying to develop a two concept basic ASRL using individual reinforcement. The leader first receives the command "all spin". The leader selects an action *straight*, and sends a signal, say **low**, to the followers. Follower 1 chooses to perform the *straight* action on the **low** signal and follower 2 chooses to perform the *spin* action. Both the leader and follower 1 receive negative reinforcement for their actions while follower 2 receives a positive reinforcement value. The leader receives negative reinforcement on the signal it broadcast since only one of the two followers performed the correct action. Note that follower 2 receives positive reinforcement relative to an ASRL signal that receives negative reinforcement. The next time the leader is given the command "all spin", it will most likely send the signal **high** since it has not explored that possibility yet. This time, follower 1 chooses to *spin* on the **high** signal and follower 2 chooses to perform the *straight* action. Follower 1 receives positive reinforcement and follower 2 receives negative reinforcement.

Follower 1 is now inclined to spin on the **high** signal while follower 2 is inclined to spin on the **low** signal. If this learning trend continues, the robots will fail to converge upon a language. The problem is that the positive reinforcement that is correct in relation to the given signal becomes problematic when the signal is negatively reinforced. In other words, the follower is learning to associate a correct output with an incorrect input — i.e. learning an invalid association. So, the follower learns a meaning for a signal that may need to change in order for all of the robots to agree on a basic ASRL.

As we will see in Chapter 4, this situation does arise in our experiments. Since the ASRL is evolving at the same time that the followers are learning their responses, individual reinforcement values that are correct at the particular moment in time may not be correct in the long run.

While individual reinforcement appears to be correct based upon the follower's action at the time it is given, it can be a "false positive" since the ASRL is still being developed — i.e. we are reinforcing the wrong association. Since the development

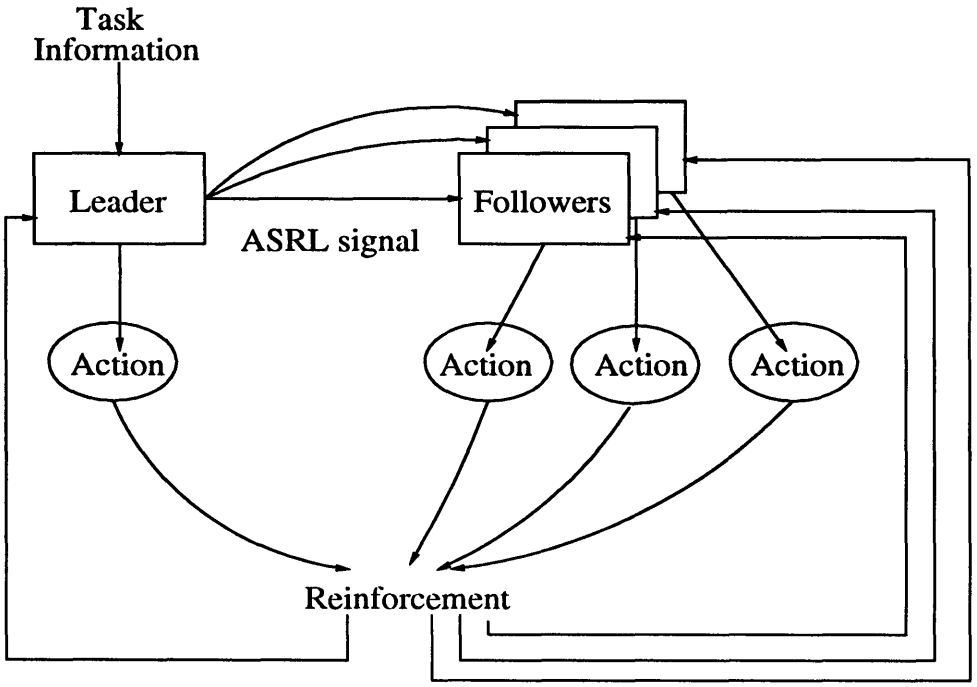



Figure 3-4: Flow of the signals in the experiments using task-based reinforcement

of ASRLs is a dependent learning problem, it is impossible to know how to reinforce the followers without the knowledge of what the leader intended; the decision as to whether a follower's action is right depends on the leader's signal. Traditional reinforcement methods do not take this into account.

### 3.3.3 Task-based reinforcement

Task-based reinforcement can handle dependent learning problems since it only provides positive reinforcement when all of the robots act properly and thus avoids the "false positive" problem of individual reinforcement. The group of robots can not receive positive reinforcement until all of the robots complete the task properly — i.e. all of the robots move together in the desired manner. The robots are reinforced relative to what will now become a stable ASRL signal. A positive reinforcement under task-based reinforcement is guaranteed to be 100% correct. A diagram of our experimental scenario using task-based reinforcement is given in figure 3-4.

Using this method, it is possible for an individual to act correctly, even relative to an appropriate ASRL signal, but still receive negative reinforcement. However, this

"false negative" does not present the same problem that the "false positive" of the individual reinforcement does. If the robot fails to receive a positive reinforcement value, it will continue to select actions randomly. Through the continued selection of random actions, the group will eventually hit upon a solution that results in positive reinforcement as long as the space of solutions is discrete and finite. If the robots have an infinite selection of actions, they might never hit upon a valid solution. Conversely, if a robot receives a "false positive" reinforcement using the individual method, the robot becomes predisposed to the selection of that action, even though it may not contribute to the ultimate global goal in which all of the robots act together.

Task-based reinforcement also provides us with the advantage of a binary reinforcement decision. If the task is completed properly, the robots are given positive reinforcement. If the task is not completed properly, the robots are given negative reinforcement. This scheme does not require any credit assignment amongst the robots.

## 3.4 Input Generalization

Most often in learning, there is a large input space; i.e. there are many inputs that can have many values. However, usually only a few of these bits are relevant. In general reinforcement learning, the algorithm explores the whole *input* $\times$ *output* space in an attempt to maximize the reinforcement that will be received (since an unexplored state may be a state with a reinforcement value far greater than anything that has been experienced). So, if there are irrelevant input bits, the algorithm will explore the extra space without making an attempt to judge whether or not it should be exploring in that area. Two of the papers that address the issue of input generalization in reinforcement learning are [Mahadevan and Connell, 1991] and [Chapman and Kaelbling, 1991].

[Chapman and Kaelbling, 1991] developed a tree-structured reinforcement table to allow for input generalization At the outset, the algorithm assumes that all input space is identical. As the learner sees examples, it builds a binary tree using input

bits that are judged to be relevant.

In [Mahadevan and Connell, 1991], they show that learning individual behaviors results in better performance than "monolithic" learning. The advantage of learning individual behaviors comes from the ability to have separate reinforcement functions for each behavior. Each of these reinforcement functions is much simpler than the function that is required for the "monolithic" global behavior. By building up smaller behaviors, we have input generalization. Each function pays attention to only relevant inputs where a global behavior would need to respond to all of the inputs (or learn to ignore certain outputs).

We did not implement any generalization algorithm in our system. We will see in the context dependent chapter that irrelevant sensor information can not be detected using our current algorithm. If the system could generalize, it could learn that only certain sensor inputs are valid. For example, it may only matter if a room is light or dark. The robot may have a large number of other sensor readings available to it, including heat sensors, touch sensors, and infrared sensors. The values from all of these other sensors should not be used to determine the meaning of a signal, since only the light value is important. In our current system, the robot must explore the space of possible sensor values — i.e. all possible combinations of sensor values.

## 3.5 Who reinforces the robots?

In order to learn, the robots require reinforcement values. These values can be provided by a human trainer, the world or another source. So long as the values are reasonably reliable, the robots will be able to learn the task. (Since most reinforcement learning algorithms can handle noise, we say "reasonably reliable".)

In the experiments on physical robots, the reinforcement was provided to the robots by a human instructor. The human instructor acted the same way as the computerized critic of the simulations. The human instructor knew the desired task for each human input and reinforced the robots based upon their actions and the reinforcement scheme in use. Because this is time consuming, most data-gathering over

large numbers of runs were performed in simulation, but the results were comparable to results on the real robots.

In our simulations, reinforcement is provided to the robots by a computerized critic. This critic knows the tasks that should be performed for each task signal given to the leader of the group. The critic uses this information to determine what reinforcement to give the robots, based upon what the robots did and the reinforcement scheme in use (in our case, either task-based reinforcement or individual reinforcement).

The experimental scenario we have constructed requires an external critic to provide reinforcement. Not even the leader knows the desired task for a given input until after it has learned by receiving reinforcement. Additionally, in the current scenario, none of the robots know what other robots in the group have done. Unless one robot had the knowledge of the proper tasks to be performed on each input and could determine what each robot did, the robots can not reinforce themselves. With the physical robots we used in the experiments (see Appendix A), it is not easy or even possible for the robots to sense what other robots have done. To simplifiy this problem, the robots could broadcast their actions to a robot that would determine the reinforcement that each robot should receive. However, giving one robot the knowledge of the correct tasks and telling it what every robot did simply turns that robot into an external critic.

The world can act as a powerful reinforcement mechanism. If the robots do not perform the correct task, they may receive the ultimate negative reinforcement: destruction. For example, if a robot needs to gather rocks from the edge of a cliff, executing the wrong movement may result in a plunge to its demise. While the world should be considered a source of potential reinforcement, we do not want to destroy robots just to avoid the requirement for an external critic. The issue regarding the type of reinforcement mechanism that should be used has been addressed in previous reinforcement learning work.

# Chapter 4

# Basic ASRL Experiments

## 4.1 Introduction

The simplest type of language that robots can develop is a one-to-one mapping of signals to actions. We call this simple language type the *basic ASRL* [1]. Portions of this chapter were joint work with Lynn Andrea Stein [Yanco and Stein, 1993].

We gain simplicity at the expense of learning times. In later chapters, we will present two other ASRL types that are more complicated and require more structure to learn, but that converge more quickly than the basic ASRL. Even though the basic language takes longer to learn, we can explore this simpler language to see how increasing the size of the robot group and increasing the number of concepts in the ASRL will affect learning time. These results demonstrate that robots can successfully develop a language.

## 4.2 Implementation

The experimental scenario was discussed in section 2.2. Recall that the leader of the group sends signals to the followers to communicate the task to be accomplished. The followers need to interpret the signal by determining the concept that maps to each

[1] Our initial experiments in the development of the basic ASRL were inspired by the work of John Shewchuk [Shewchuk, 1991]

signal. The learning problem is to match the signals to concepts. In our experiments, concepts map directly to actions due to Assumption 6 in Chapter 2. The ASRL evolves as the leader and followers are learning the task.

Real and simulated robots have developed the basic ASRL. Our statistical data collection was done in simulation. Since the scenario we have set up does not depend on the world at all, the results using the simulation are comparable to results using real robots. While operating in the world usually results in some amount of noise (for example, communication noise), the assumptions we made in section 2.1 guarantee that real-world operation and simulation are comparable in these experiments.

## 4.3 Results and Discussion

An example of the development of a two concept basic ASRL by two robots is given in table 4.1. The two task signals given to the leader are ○○ and ↑↑ . On input ○○ , both robots should spin in place. On input ↑↑, both robots should go straight. The leader has a set of two initially uninterpreted signals to send to the follower: **high** and **low**. Each robot selects from two possible actions to perform: *straight* and *spin*. Using task-based reinforcement, the robots either receive positive reinforcement, +, or negative reinforcement, −. Task-based reinforcement only reinforces the correct total action as can be seen in line 2 of this table. The leader executed the proper action, *spin*, but it receives negative reinforcement since the entire task was not completed correctly because the follower chose the action *straight*. After twelve iterations, the robots converge upon an ASRL where **low** maps to the concept *spin* and **high** maps to the concept *straight*.

Table 4.2 shows an example of the development of a two concept basic ASRL by three robots using task-based reinforcement. The task signals are ○○○ meaning "all spin" and ↑↑↑ meaning "all straight." It takes longer for three robots to learn the two concept language because there is now another robot that must act correctly in order for the robots to receive positive reinforcement using the task-based reinforcement method. For example, in line 5 of table 4.2, the leader and follower 1

| | Appropriate action | Leader's action | Leader's signal | Follower's action | Reinforcement |
|---|---|---|---|---|---|
| 1. | ○○ | *spin* | low | *straight* | − |
| 2. | ○○ | *straight* | high | *spin* | − |
| 3. | ↑↑ | *straight* | high | *straight* | + |
| 4. | ○○ | *spin* | low | *spin* | + |
| 5. | ○○ | *spin* | low | *spin* | + |
| 6. | ○○ | *spin* | low | *spin* | + |
| 7. | ↑↑ | *spin* | low | *spin* | − |
| 8. | ○○ | *spin* | low | *spin* | + |
| 9. | ↑↑ | *straight* | high | *straight* | + |
| 10. | ↑↑ | *straight* | high | *straight* | + |
| 11. | ↑↑ | *straight* | high | *straight* | + |
| 12. | ○○ | *spin* | low | *spin* | + |

Table 4.1: A sample run demonstrating the development of a two concept basic ASRL by two robots using task-based reinforcment. The desired behavior is *both spin* on input ○○, *both go straight* on input ↑↑. After twelve iterations, convergence is reached. The ASRL agreed upon by the two robots is low → *spin* and high → *straight*.

both choose *straight*. In the previous example with two robots, these outputs would have resulted in positive reinforcement. However, since the third robot, follower 2, chooses *spin*, the group receives negative reinforcement. After 24 iterations, the robots converge upon an ASRL where low maps to the concept *straight* and high maps to the concept *spin*.

### 4.3.1 Development of different dialects

As would be expected, the robots develop different mappings for words to actions in different runs due to the initial random selections of actions. These different mappings can be considered different dialects. In table 4.1, the robots agreed upon a basic ASRL in which low mapped to *straight* and high mapped to *spin*. In table 4.2, the robots converged upon a basic ASRL in which low mapped to *spin* and high mapped to *straight*. Robots trained separately that develop different dialects would not be able

| | Appropriate action | Leader's action | Leader's signal | Follower 1's action | Follower 2's action | Reinforcement |
|---|---|---|---|---|---|---|
| 1. | ↑↑↑ | *straight* | low | *straight* | *spin* | − |
| 2. | ↑↑↑ | *spin* | high | *spin* | *straight* | − |
| 3. | ↑↑↑ | *spin* | high | *straight* | *spin* | − |
| 4. | ↑↑↑ | *straight* | low | *spin* | *straight* | − |
| 5. | ↑↑↑ | *straight* | low | *straight* | *spin* | − |
| 6. | ○○○ | *spin* | high | *spin* | *spin* | + |
| 7. | ↑↑↑ | *spin* | high | *spin* | *spin* | − |
| 8. | ↑↑↑ | *straight* | low | *spin* | *straight* | − |
| 9. | ↑↑↑ | *spin* | high | *straight* | *straight* | − |
| 10. | ↑↑↑ | *spin* | high | *spin* | *spin* | − |
| 11. | ↑↑↑ | *straight* | low | *straight* | *spin* | − |
| 12. | ○○○ | *straight* | low | *spin* | *straight* | − |
| 13. | ○○○ | *spin* | high | *spin* | *spin* | + |
| 14. | ○○○ | *spin* | high | *spin* | *spin* | + |
| 15. | ↑↑↑ | *straight* | low | *spin* | *spin* | − |
| 16. | ↑↑↑ | *spin* | high | *spin* | *spin* | − |
| 17. | ↑↑↑ | *spin* | high | *spin* | *spin* | − |
| 18. | ○○○ | *spin* | high | *spin* | *spin* | + |
| 19. | ↑↑↑ | *straight* | low | *straight* | *straight* | + |
| 20. | ↑↑↑ | *straight* | low | *straight* | *straight* | + |
| 21. | ○○○ | *spin* | high | *spin* | *spin* | + |
| 22. | ↑↑↑ | *straight* | low | *straight* | *straight* | + |
| 23. | ○○○ | *spin* | high | *spin* | *spin* | + |
| 24. | ↑↑↑ | *straight* | low | *straight* | *straight* | + |

Table 4.2: A sample run of the development of a two concept ASRL by three robots (one leader and two followers) using task-based reinforcement. The desired behavior is *all spin* on input ○○○, *all go straight* on input ↑↑↑. After twenty-four iterations, convergence is reached. The robots agree upon the ASRL in which low → *straight* and high → *spin*.

to effectively communicate with one another. However, since the learning of different dialects is typical of independent populations, we would expect to see this result.

### 4.3.2 Adaptability of language

We have performed experiments on the robots in which we change the meaning of task inputs after the robots have learned a language. For example, after the robots have learned to perform the task "all spin" on human input 1 and the task "all straight" on human input 2 correctly, we change the desired behavior for human input 1 from "all spin" to "leader spin, followers straight". Since the command is simply a monolithic idea that can not be broken down into smaller parts, the robots will try to continue to execute the old behavior for the input until enough negative reinforcement has been received to force the robots to try new actions. If we implement a memory window in our reinforcement learning algorithm, the robots would only remember a given number of trials. Alternatively, we could weight reinforcement values with the most recent reinforcement values weighted more heavily than older reinforcement values. This allows for relearning to occur much more expediently than it would if the meaning of a signal changed after 1,000,000 correct trials in which a large amount of positive reinforcement would have been collected.

In our robotic trials, we changed the meanings of signals very soon after the robots had learned the original meanings, so that using a memory window or discounted rewards was not necessary. Had the robots been working in the real world instead of controlled experiments, we would have needed a memory window or another method to allow the robots to relearn in a reasonable amount of time. Other methods could throw out previously learned information after a given number of unexpected reinforcement values since the surprise would cause the robots to want to relearn the proper behavior or could use large negative values for negative reinforcement to quickly decrement the built up reinforcement for a behavior.

An example of this relearning behavior in a robotic trial is given in table 4.3. The two robots need to learn to both spin on ◯◯ and to both go straight on ↑↑. The robots converge after twelve iterations to an ASRL where low maps to *spin* and high

maps to *straight*. On step 13, the task signal ↑↑ is changed to mean "both spin." We denote this in the table as ↑↑→ ◯◯. In steps 13–18, the robots are adapting to this change. The solution involves using the **high** signal which was previously mapped to *spin* in the ASRL. It provides a quick solution since the followers already had learned to *spin* on the **high** signal. For this adaptation, the leader learns to say something different. Then, in steps 19–34, the task signal ◯◯ is changed to mean "both straight." The solution this time is to remap the **high** signal in the language to mean *straight* — the followers learn to interpret the signal differently, providing the language with shifting meanings. Once this remapping occurs, the solution found in steps 13–18 for ↑↑→ ◯◯ is no longer valid. The robot will need to adapt further if the ↑↑→ ◯◯ command is given again with the same expected output. This example shows two ways that the robots are able to adapt. It also shows that long strings of one input can cause learning problems; without competition, one signal can flip back and forth to two concepts while the other signal goes unused.

### 4.3.3 Adding more concepts to the ASRL

In learning problems, the addition of new concepts to be learned increases the time required to reach convergence. To explore what happens as we scale up the language size, we have run experiments in which we vary the number of concepts in a language from two to twenty using two and three robots.

A comparison between two robot and three robot development of basic ASRLs varying in size from two to twenty concepts using individual reinforcement is graphed in figure 4-1 and the data is given in tables B.3 and B.4. A comparison between two robot and three robot development of basic ASRLs varying in size from two to twenty concepts using task-based reinforcement is graphed in figure 4-2 and the data is given in tables table B.1 and B.2. The amount of time necessary to learn is sub-exponential in the number of concepts in the language.

Due to these quickly increasing learning times, we will explore other language types that try to overcome some of these problems. In Chapter 5, we discuss the development of a context dependent ASRL. In Chapter 6, we discuss the development

| | Appropriate action | Leader's action | Leader's signal | Follower's action | Reinforcement |
|---|---|---|---|---|---|
| 1. | ○○ | *straight* | low | *straight* | − |
| 2. | ↑↑ | *straight* | low | *spin* | − |
| 3. | ○○ | *spin* | high | *straight* | − |
| 4. | ↑↑ | *spin* | high | *spin* | − |
| 5. | ↑↑ | *straight* | low | *straight* | + |
| 6. | ○○ | *spin* | high | *straight* | − |
| 7. | ↑↑ | *straight* | low | *straight* | + |
| 8. | ↑↑ | *straight* | low | *straight* | + |
| 9. | ○○ | *straight* | low | *straight* | − |
| 10. | ○○ | *spin* | high | *spin* | + |
| 11. | ○○ | *spin* | high | *spin* | + |
| 12. | ○○ | *spin* | high | *spin* | + |
| 13. | ↑↑→ ○○ | *straight* | low | *straight* | − |
| 14. | ↑↑→ ○○ | *straight* | low | *straight* | − |
| 15. | ↑↑→ ○○ | *straight* | low | *spin* | − |
| 16. | ↑↑→ ○○ | *spin* | high | *spin* | + |
| 17. | ↑↑→ ○○ | *spin* | high | *spin* | + |
| 18. | ↑↑→ ○○ | *spin* | high | *spin* | + |
| 19. | ○○→ ↑↑ | *spin* | high | *spin* | − |
| 20. | ○○→ ↑↑ | *spin* | high | *spin* | − |
| 21. | ○○→ ↑↑ | *spin* | high | *spin* | − |
| 22. | ○○→ ↑↑ | *straight* | high | *spin* | − |
| 23. | ○○→ ↑↑ | *spin* | low | *straight* | − |
| 24. | ○○→ ↑↑ | *straight* | high | *spin* | − |
| 25. | ○○→ ↑↑ | *spin* | high | *straight* | − |
| 26. | ○○→ ↑↑ | *spin* | low | *straight* | − |
| 27. | ○○→ ↑↑ | *straight* | high | *spin* | − |
| 28. | ○○→ ↑↑ | *spin* | high | *spin* | − |
| 29. | ○○→ ↑↑ | *straight* | low | *straight* | + |
| 30. | ○○→ ↑↑ | *spin* | high | *spin* | − |
| 31. | ○○→ ↑↑ | *straight* | low | *straight* | + |
| 32. | ○○→ ↑↑ | *straight* | high | *straight* | + |
| 33. | ○○→ ↑↑ | *straight* | high | *straight* | + |
| 34. | ○○→ ↑↑ | *straight* | high | *straight* | + |

Table 4.3: A sample run demonstrating the development and adaptation of a two concept basic ASRL by two robots using task-based reinforcement. This figure demonstrates two ways that the robots can adapt.

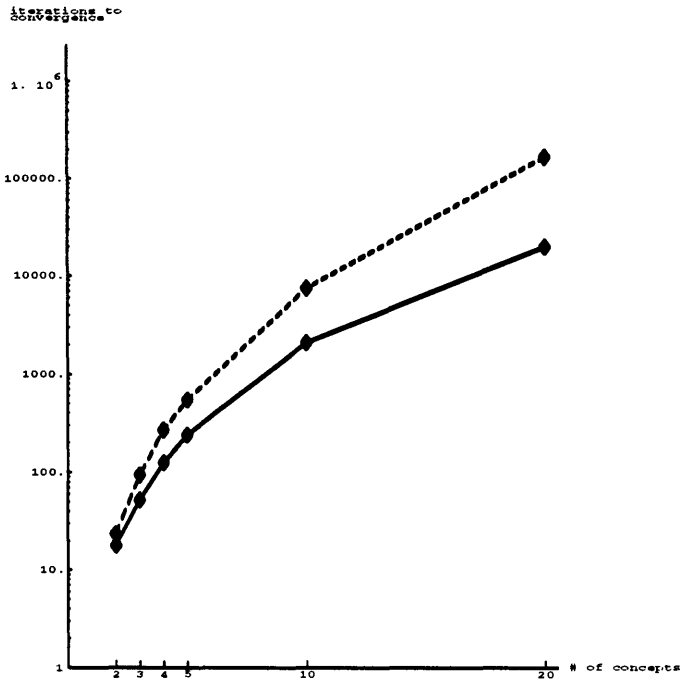



Figure 4-1: Comparison of running times for the development of basic ASRLs using individual reinforcement. The solid line graphs results using two robots and the dashed line graphs results using three robots.

of a compositional ASRL. Both of these language types use some built-in structure to produce better learning times.

### 4.3.4 Task-based vs. individual reinforcement

The robots have developed basic ASRLs using both task-based reinforcement and individual reinforcement. As mentioned previously, task-based reinforcement is better suited to the dependent learning problem the robots face in the experimental scenario.

In individual reinforcement, each robot receives a reinforcement value that depends on its action in relation to the task that needs to be completed. Consider the task of "all spin". An individual robot receives positive reinforcement when it spins, since it is acting properly towards the completion of the group task. In our scenario, the robots are evolving their language as they learn to complete the task. The ASRL signal is reinforced in a task-based manner since the signal is not a good one unless all of the followers act in the proper manner.

For the two robot case, there are no convergence problems. Since there is only

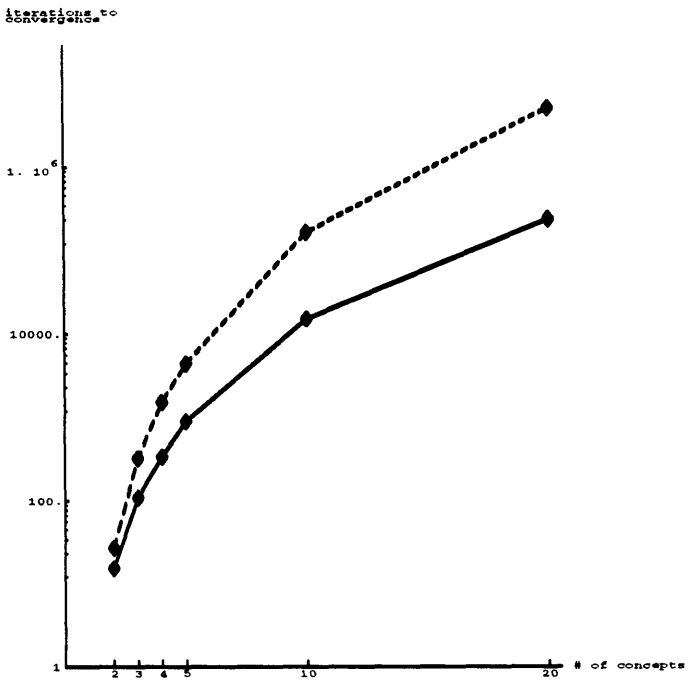



Figure 4-2: Comparison of running times for the development of basic ASRLs using task-based reinforcement. The solid line graphs results using two robots and the dashed line graphs results using three robots.

one follower, there is no possibility of falling into an alternating and non-converging scenario as described in section 3.3.2.

However, we see in the three robot case that convergence problems are very real. To collect 100 runs for our average 20 concept basic ASRL data point, we needed to throw away 445 runs; i.e. only 100 of 545 runs converged. Clearly, this is unacceptable. We want the robots to learn a language as quickly as possible so that they will be working reliably as fast as they can; however, the apparent speed up in learning times for individual reinforcement over task-based reinforcement is inconsequential when we must throw away four-fifths of the runs as non-converging. A comparison of the learning times for the development of basic ASRLs by three robots using task-based reinforcement vs. individual reinforcement is given in figure 4-3 and data is given in tables B.3 and B.4.

Individual reinforcement is difficult to use when tasks can not be decomposed as easily as our example of "all spin." For example, if the task is "move the couch", how do we determine which of the robots actually helped move the couch? Perhaps

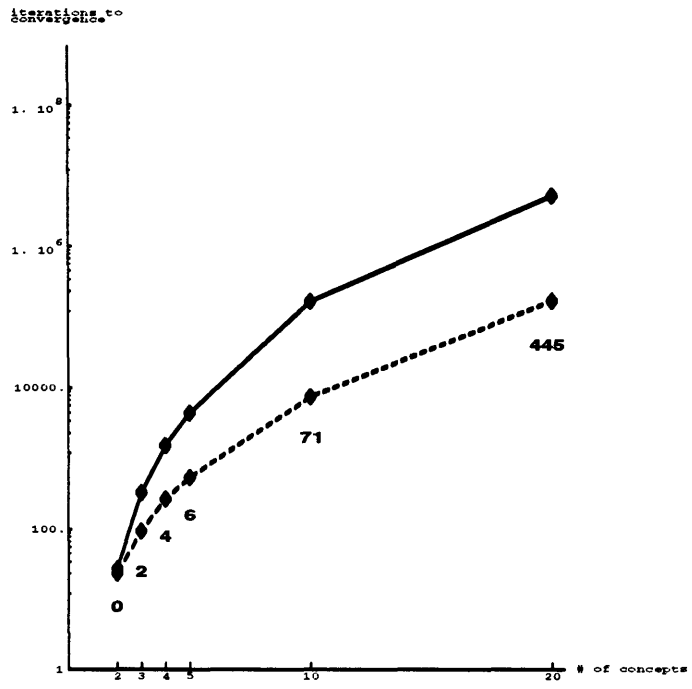



Figure 4-3: Comparison of running times for task-based reinforcement and individual reinforcement in the development of basic ASRL by three robots. The solid line graphs results of basic ASRL development using task-based reinforcement. The dashed line graphs results using individual reinforcement. The numbers next to the individual reinforcement points give the numbers of non-converging runs that needed to be thrown away in order to collect 100 converging runs for our average.

we should just reward any robot that touches the couch since they are attempting to help. A robot on the wrong side of the couch may be preventing movement or it could actually be helping to balance inequities in the pushing capabilities of the robots on the other side of the couch. Even if it's difficult to determine which of the robots that come in contact with the couch assist in the completion of the task, it should be easy to say that robots that did not touch the couch did not help, right? Wrong. A robot may remove itself from the pushing task if it knows it would do more harm than good by attempting to assist. It may also remove itself from the task in order to move away an obstacle blocking the path that the other robots will use to move the couch. Moving the couch is a dependent problem; the success of one robot depends upon the action of other robots.

The advantage of task-based reinforcement is that it can handle the problems of individual reinforcement. In the couch moving problem, the robots only receive

positive reinforcement if the couch is moved. Even if one or more of the robots did not contribute to the goal directly, the task of moving the couch was accomplished. If these robots were to do the same thing again, the couch would be moved again (in the ideal world where robots don't break down).

Task-based reinforcement also works in our experimental scenario. The followers only receive positive reinforcement when everyone acts correctly; therefore, they have no possibility of falling into the non-convergence problem described above in the discussion of individual reinforcement. We have seen empirically that the task-based reinforcement method results in convergence for every run. This empirical evidence agrees with the intuition that the robots can not fall into the degenerate state that can be caused by individual reinforcement.

### 4.3.5 All robots have knowledge of other robots' actions

What if all of the robots knew what the others had done on a given command? Individual reinforcement would not benefit from this knowledge since each robot is given its own reinforcement value based upon its action. Knowledge is useful to help determine why a particular action was incorrect; in the case of individual reinforcement, the non-convergence problem occurs due to "false positive" reinforcement values so this would not help solve that problem.

However, in the case of task-based reinforcement, it seems that it would speed up learning if the robots interpreted their reinforcement with the knowledge of what other robots in the group had done. It seems easy to say, "Well, I think I acted properly, but received negative reinforcement. However, since someone else may have acted wrong, maybe I actually did the right thing." The robots are not able to deduce this easily. In order for them to incorporate knowledge of what the other robots did, we need to add an extra dimension to the learning tables. This extra dimension represents what the other robots have done. So now the tables are $(inputs \times actions\_of\_other\_robots) \times outputs$.

So, does this speed up learning? No. Adding extra dimensions to a learning table only slows down the learning process. The learning algorithm can not make

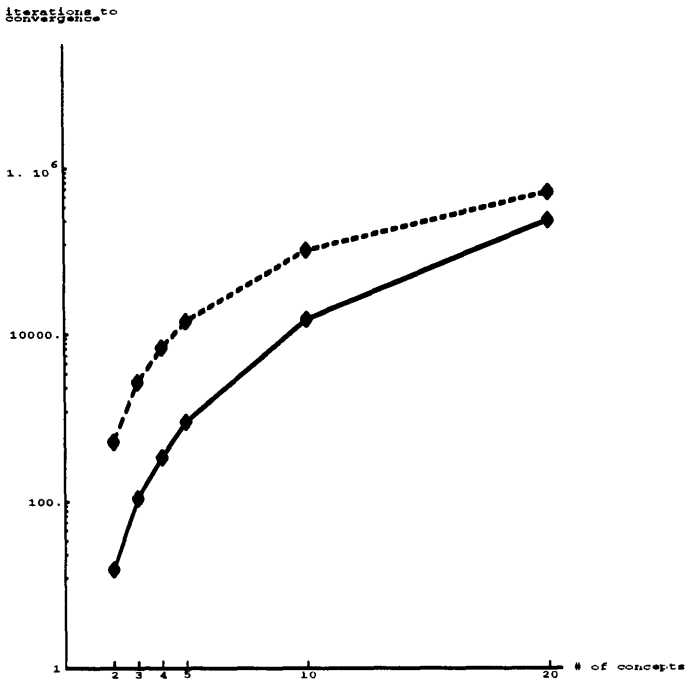



Figure 4-4: Comparison of running times for the development of a basic ASRL by two robots with and without knowledge of what the other robot is doing using task-based reinforcement. The solid line graphs results using no knowledge and the dashed line graphs results using knowledge.

generalizations across the table; it needs to see a situation before it can generate the proper response. Therefore, added inputs to the table just increase the amount of time it takes the robot to explore the possible states. The result of developing the ASRL with knowledge vs. developing without knowledge for two robots is graphed in figure 4-4 and the data points are given in tables B.1 and B.5.

### 4.3.6 Scaling up the group size

In our scenario, adding extra robots to the group also increases learning times. To explore what happens as we scale up the group size, we have run experiments in which we vary the the team size from two to eight robots. While we have used groups of two and three robots in the real world, we have run these experiments in simulation where we have an infinite supply of robots that will not break down.

When we vary the group size from two to eight robots for language sizes of three and four elements, we see that adding an extra robot exponentially increases the

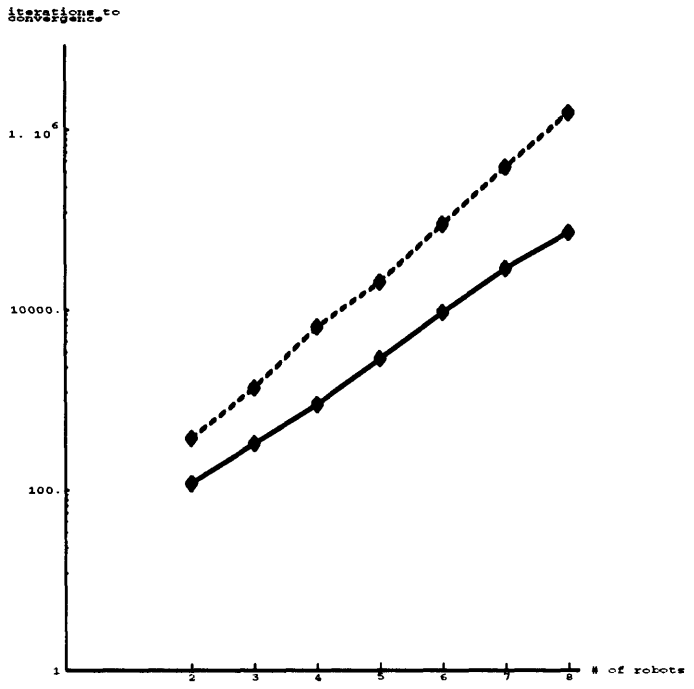



Figure 4-5: Comparison of running times for basic experiment with group sizes varying from two to eight robots. The solid line represents data for the development of three concept basic ASRLs. The dashed line represents data for the development of four concept basic ASRLs.

amount of time necessary to learn the task. The results of these experiments are graphed in figure 4-5. and The values for the points are given in tables B.6 and B.7. These results were gathered using task-based reinforcement.

## 4.4 Future work

Other reinforcement learning methods should be implemented and tested in our framework. This will help us to determine whether or not task-based reinforcement is applicable to all methods of reinforcement learning. Additionally, other reinforcement learning algorithms may be better suited to the language learning task and result in faster learning times.

For interval estimation, we could speed up the exploration phase of the algorithm by using the knowledge that only one of the actions will result in a positive reinforcement value for a given input. Once we receive a positive reinforcement value,

we could immediately begin to exploit that knowledge and cease exploring all of the other actions. The exploration of all concepts is a limiting factor in the learning as we scale up to languages with larger numbers of concepts. However, we do not want to eliminate exploration altogether because the world is dynamic and other outputs may return larger rewards.

This knowledge that we have one-to-one mappings is only useful for task-based learning since we know that positive values given by task-based learning are reliable. Positive values given using individual reinforcement may be correct at the time of reinforcement, but may change as the language evolves. If we were to use the one-to-one mapping extension with individual reinforcement, one positive reinforcement value could keep the robots from ever converging upon a language (as opposed to the situation now where at least some of the runs converge).

Another area for future work is bootstrapping. The robot language could be developed by a small number of robots to avoid the exponential blow up in learning times for each additional robot that is trying to concurrently learn. Once a small group of robots develops the ASRL, it can be transferred to new robots joining the team, either by a group member training the new member or by directly reading the reinforcement learning tables from an existing team member to the new member.

# Chapter 5

# Context Dependent Language Experiments

## 5.1 Introduction

A word can have many meanings in a context dependent language. Using context when determining meaning allows for the use of fewer words in a language to represent a given number of concepts. We will discuss later in this chapter why it is advantageous to have smaller languages. There are two types of context that can be used to determine meanings of words – *sentence-based context* and *state-based context.*

In sentence-based context dependency, the meaning of a word depends on the context of the surrounding sentence. For example, the word *watch* has two different meanings in the sentence "Don't watch your watch."

In state-based context dependency, which is also known as indexicality, the meaning of a word depends on the state of the world in which it is spoken. For example, in the simple command "Look at this", the word "this" can have a variety of meanings. The object that the speaker is referencing determines the meaning of the word "this." In order for the listener to understand the meaning of the speaker's sentence, the listener must be in the same context as the speaker or must be able to move into that context. If the speaker says "Look at this" while pointing to an object hidden from the listener by a desk, the listener needs to ask for more information or needs

to get up and move to where he can see the object being referenced.

The work in this chapter addresses state-based context dependency. (We will discuss sentence-based context dependency in terms of our compositional language development in Chapter 6.)

In this work, a collection of simulated robotic agents are being trained to develop a context dependent language. The agents could learn a command such as "Do", where the meaning of "Do" is directly mapped by the sensor readings to a set of appropriate concepts. For example, agents could do "lunch" where there is food, do "gather objects" where objects are present and do "avoid" when predators that are detected. In this case, "Do" would be mapped by the sensor readings {food present, object present, predator present, ...} to the concepts {eat food, gather object, avoid predator, ...}. Another command could be "Move away", where "Move away" is mapped by the sensor readings {light, heat, ...} to the concepts {go to dark, go to cold, ...}.

The advantage of a context dependent language is the need for fewer signals to represent concepts. In a robotic system where the number of possible communication signals is bandwidth limited, the number of concepts that can be represented increases as more sensor data is used as a disambiguating context. The number of concepts that can be represented is given by

$$num\_concepts = num\_signals[\prod_{i=1}^{num\_sensors} num\_sensor\_values_i]$$

where $num\_signals$ is the number of signals that the robots can send and $num\_sensor\_values_i$ is the number of values that $sensor_i$ can have.

## 5.2 Implementation

The experiments performed are based upon the basic language learning experiment described in the previous chapter. Two agents learn to communicate with each other in the presence of a trainer. The experimental scenario is the same as the one de-

| Number of | | | Number of Iterations to Convergence | | |
|---|---|---|---|---|---|
| Signals | Sensor Values | Concepts | Average | Minimum | Maximum |
| 2 | 2 | 4 | 219.34 | 68 | 555 |
| 2 | 3 | 6 | 871.84 | 297 | 2676 |
| 2 | 4 | 8 | 2287.57 | 883 | 5815 |
| 2 | 5 | 10 | 5165.33 | 1887 | 12600 |
| 2 | 10 | 20 | 51860.30 | 21853 | 105632 |

Table 5.1: Data from experiments with two signal context dependent ASRLs. In order to increase the number of concepts that the language can describe, we increase the number of sensor values that are used in conjunction with a signal to determine meaning. Each of the languages developed will be optimal; i.e. the fewest signals are used to encode the maximum amount of data.

scribed in section 2.2. We used task-based reinforcement in these experiments.

In the context dependent extension to the experiment, the agents take into account a sensor-reading vector when considering the meaning of a signal. Both task signals and ASRL signals are interpreted using sensor-reading vectors. Once again, the agents need to map signals (or words) to concepts (or meanings for the words). To motivate the agents to create multiple context dependent mappings from signals to concepts, the language size is restricted; i.e. there are not enough signals for the robots to create a one-to-one mapping between signals and concepts. We have also performed experiments where the language size is larger than necessary; this is discussed below in the results section.

Currently, the use of sensor values in the determination of meaning is built-in because we have incorporated the sensor values into the reinforcement learning table of *inputs* $\times$ *outputs*. The sensor values are paired with the signals and used as the input.

In these experiments, it is assumed that the agents are communicating in a common context. Since the work was done in simulation, keeping the robots in the same sensor context was simple.

## 5.3 Results and Discussion

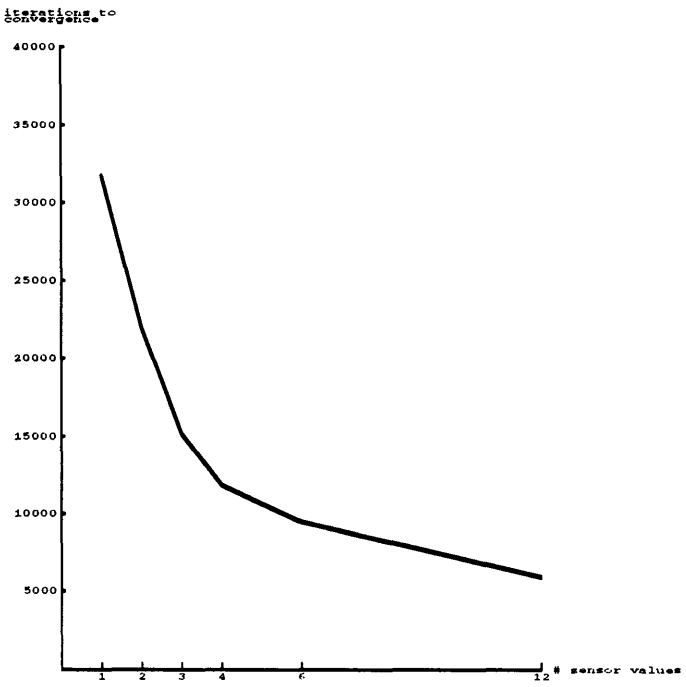



Figure 5-1: As more sensor values are used to interpret signals, learning time decreases.

We have performed various experiments using the simulation described above. In table 5.1, learning times are given for the case where the agents have two signals that are mapped using an increasing number of sensor values. In this experiment, the agents are learning the *optimal* language, which we define to use the smallest number of words to encode the desired number of concepts.

The advantage of a context dependent language is that using sensor values together with the ASRL signals allows us to use fewer signals in the ASRL where in the basic language we needed one signal for each concept. We ran experiments for the development of 12 concept languages using 1 to 12 sensor values to determine the meaning of signals. We kept the language optimal; i.e. there were $12/num\_sensor\_values$ signals provided to the leader. As the robots need to learn fewer ASRL signals, the learning times decrease. The results of these experiments are graphed in figure 5-1 and the values for the points are given in table B.8.

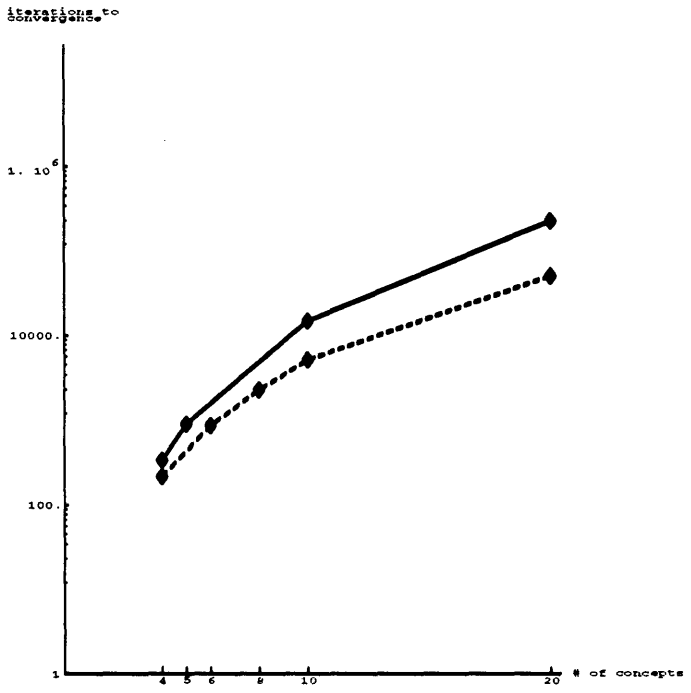



Figure 5-2: Comparison of learning times for context dependent ASRLs and basic ASRLs. The dashed line graphs results for the development of context dependent ASRLs and the solid line graphs results for the development of basic ASRLs.

### 5.3.1 Context dependent ASRL vs. basic ASRL

A graph comparing learning times for the development of basic ASRLs and context dependent ASRLs is given in figure 5-2. The points graphed in this figure are given in tables B.1 and 5.1. Note that the context dependent ASRL can be developed in fewer iterations than that basic ASRL. This speed-up in learning is due to the smaller reinforcement table required for the leader to learn the appropriate signal to send to the followers. For example, in the basic experiment, if we want the leader to be able to send signals for ten different concepts, the table for learning the task signal $\times$ ASRL signal is $10 \times 10$. In the context dependent work, the number of signals the robot needs to send is reduced by a factor of the number of sensor values being used to interpret the meaning of a signal, which reduces the learning times.

### 5.3.2 Non-optimal languages

When the agents have a number of signals for their language that is greater than

| Number of | | | | Number of Iterations to Convergence | | |
|---|---|---|---|---|---|---|
| Task Signals | Sensor Values | ASRL Signals | Concepts | Average | Minimum | Maximum |
| 2 | 4 | 2 | 2 | 91.26 | 60 | 152 |
| 2 | 2 | 4 | 4 | 175.27 | 64 | 460 |
| 2 | 4 | 2 | 4 | 556.49 | 155 | 1677 |

Table 5.2: Learning times for cases with extra values. In the first line, the four sensor values are not necessary; the problem has a solution using only two human signals, two robot signals and two actions. In the second line, the agents have a choice of four signals for their language where only two are required for an optimally sized language. In the third line, only two of the four sensor values are significant. Each experiment was run 100 times on a troupe size of two agents.

the optimal size, it actually takes less time to converge than the optimal language size. We performed an experiment where we gave the agents four signals to represent a language that only required two signals (with two sensor values, mapping to four concepts). It took an average of 175.27 trials vs. the average of 219.34 for the case where the robots were given two signals and two sensor values for four concepts. This happens because the larger language size allows the agents to randomly select signals with fewer collisions (defined as a selection that can not co-exist with a selection previously made) than in the case where each signal needs to be mapped as many times as it can be (as determined by the number of sensor values).

When given four signals to create their language, the agents created a two signal language 16 times, a three signal language 70 times and a four signal language 14 times over 100 runs. In order for the robots to create an optimal language in the presence of additional signals, we would need to place a learning restriction that favors smaller languages.

When the sensor data is irrelevant, the robots will still converge upon a solution. However, it takes longer than the case where no sensor data is given; a two robot, two signal, two action basic ASRL experiment described in the previous chapter has an average convergence time of 15.24. In an experiment with two agents, two signals, two actions *and two sensor values*, the average convergence time is 91.26. Even though the sensor data is irrelevant, each robot must learn the proper action for

each $(signal, sensor)$ pair; i.e. it must fill in the table for the reinforcement learning algorithm. In order for the learning to be more efficient, the agents will need to detect irrelevant sensor values and ignore them. The detection of relevant information, or input generalization, has been explored by researchers in reinforcement learning. We discussed input generalization in Chapter 3.

Another experiment that we have tested is the case where there are four sensor values representing two bits of sensor data. If only one bit is significant, the convergence time is still based upon the fact that there are four sensor values. For two signals, four sensor values and four actions, the average convergence time is 556.49. Once again, we see that the agents are not able to ignore unnecessary input.

## 5.4 Future Work

The experiments described in this chapter have only been performed in simulation. The next step is moving them to real robots. Context would be provided by actual sensor readings on the robots rather than from a simulated sensor vector. Different areas in the world would have various characteristics that the robots would be able to detect with their sensors. The robots have sensors that detect light levels, the presence of objects, heat levels and infrared signals. Due to the noise inherent in sensor readings, we expect that learning times will be slightly larger on the actual robots.

Additionally, the assumption that the communicating robots are in the same context will not be valid for real robots. The listener robot will need to figure out what context the speaker robot is in. One way to do this is to require a robot to broadcast the sensor information in the same packet as the information that is being sent. The robots would either need to learn to interpret the context information as it is learning the language or the robots would need to be provided with the means for understanding the context broadcast that precedes the signal. This raises the issue of what a robot should do with a signal coming from a robot in another context.

Sending context increases the length of the signal that needs to be sent. In prac-

tice, shorter signals usually are more reliable. However, since we have assumed perfect communication, the longer message is as reliable as the shorter message without context information. However, we may run into bandwidth problems.

Sending context will affect learning. Should the robots use their context to determine the meaning of a signal or use the sender's context? Should a robot simply ignore any message preceded by a context that does not match its own? Since sensor values in the world do not produce reliable, discrete values, the robots may never have contexts that agree completely.

The learning algorithm we are using requires an exploration of the complete space. When there are many irrelevant sensor values, the robots will blindly search to try to find the best reinforcement value possible. Input generalization could help us to overcome this problem. This algorithm attempts to determine which of the input bits are relevant to the problem. Prior work in input generalization was discussed in Chapter 3.

# Chapter 6

# Compositional Language

## 6.1 Introduction

When new concepts are introduced in the basic language, the robots must learn the concept from scratch. This tabula rasa learning method causes learning times for each new concept to be long, even if the new concept is a variation of a previously learned concept; in fact, the learning time is sub-exponential in the number of concepts in the basic ASRL. What can we do to the language to allow for the use of previously learned concepts in deciphering a new concept?

Consider the language in which this thesis is written. Each sentence can be thought of as representing a concept. This particular sentence probably contains an ordering of English words that you have not experienced yet in all of your years of reading English; however, you are able to understand this sentence by using your knowledge of the meanings of individual words and your understanding of how words can be put together to form sentences.

ASRLs can benefit from this observation about human language. Instead of forcing the robots to learn unseen concepts from scratch, we could give them the capability to use portions of previously learned concepts to interpret these new concepts. This is the power of a compositional language in which utterances can be decomposed into their component words.

We introduce a simple, fixed position grammar that the robot uses to learn a

compositional ASRL. Note that the robot does not create this simple grammar; it is provided to the robot. While we need not look to humans for justification, Chomsky [Chomsky, 1986] argues persuasively that humans are born with an innate grammar; i.e. children do not need to learn most of the structure of a language before starting to learn the language. We discuss the grammar we implemented in the next section.

In the previous chapter, we gave two definitions of context dependency. In one of the two types, the world state is used to determine the context in which the word should be interpreted. In the other type, the surrounding words in a sentence provide context. In these experiments, we have not used the context of surrounding words to help determine the meaning of a word since we have a fixed grammar. If we were to switch to a non-fixed grammar, the relationship of a word to other words would be much more important. However, we are using a small amount of context dependency. The meaning of a word depends upon the word slot it appears in; this interpretation of categorization is built in to our system.

We present results that show that adding the bias of a grammar into the learning of a compositional ASRL allows the robot to learn a language more quickly than a basic ASRL. To compare the two types of ASRLs, we think of the ASRLs in terms of the number of concepts that they can represent rather than the number of words in the language. In the basic ASRL, the number of words and the number of concepts are equal. However, in the compositional ASRL, the number of concepts can be much larger than the number of words in the ASRL. The number of concepts is equal to the product of the number of words for each word slot over the number of word slots, i.e

$$num\_concepts = [\prod_{i=1}^{num\_slots} num\_words_i + 1] - 1$$

We add one to each of the number of words in a slot since a slot may be left empty and subtract one from the number of concepts since a blank expression is not really a concept. This is discussed further in the implementation section.

For example, consider a case where we have a three slot grammar. The first

slot can either contain *spin* or *straight* or be blank. The second slot can be set to *fast*, *slow* or blank. The last slow can be set to *short*, *long* or blank. We can form $[(2+1)(2+1)(2+1)]-1 = 26$ concepts using two words in each of the three slots. A few of these possibilities are "spin fast long", "straight slow long", "spin (blank) short", and "straight fast (blank)".

Compositional ASRLs that have even distributions of words over word slots will be able to represent more concepts than compositional ASRLs with uneven distributions. For example, let's consider a compositional ASRL with 3 word slots and 12 words in the ASRL. If the words are evenly distributed amongst the word slots (i.e. there are 4 words that can be in each slot), the ASRL can represent $[(4+1)(4+1)(4+1)]-1 = 124$ concepts. If one slot had 10 of the words and each of the other slots only had one, the ASRL would only be able to represent $[(10+1)(1+1)(1+1)]-1 = 44$ concepts. In the results and discussion section of this chapter, we will see that even distribution of words over word slots not only maximizes the number of concepts that can be represented but also minimizes the learning time.

## 6.2 Implementation

The learning scenario in these experiments differs from the experimental scenario described in section 2.2. In both the basic language and context dependent language experiments, a task signal would be given to the leader of the robot group and the leader needed to evolve a language that it and its followers could understand in order to perform the necessary tasks. In these compositional language experiments, one simulated robot learns to understand compositional task signals.

We use a single robot since our original experimental scenario does not handle the compositional ASRL well. If the human gives the leader a compositional signal, what should the leader do with it? Should it simply learn a one-to-one mapping from slots in the human command to slots in the robot ASRL? If the human only provides a monolithic signal, the leader has no way to determine if the concept is similar to a previously learned concept. In order for the work to be extended to multiple robots,

we believe that a different experimental scenario is required; for now, we simply use one robot.

The single robot in these experiments can be thought of as a follower in the original scenario, where the incoming signals would be coming from another robot. To allow for accurate comparisons, we also reimplemented the simulations of the development of the basic ASRL with only one robot.

In our simulation, the robot needs to interpret an incoming signal using a built-in grammar. The grammar that we implemented is a three word, fixed position grammar. Using fixed position allows the robot to know which category a particular word fits into. The robot could learn this categorization in addition to learning the meaning of the words if we implemented a non-fixed grammar; using a fixed grammar provides for faster learning since it would take exponential time proportional to the number of categories to learn the categorizations of the words.

We implemented the compositional ASRL learning using phrase reinforcement and word reinforcement. Phrase reinforcement is like task-based reinforcement. Using this reinforcement method, the robot only receives positive reinforcement if it sets all of the parameters correctly for all of the words in the phrase. Word reinforcement is like individual reinforcement where the robot receives positive reinforcement for each parameter it sets correctly given the corresponding word in the sentence. Since we allow words to be blank, phrase reinforcement can sometimes look like word reinforcement.

We will talk about the concepts that can be represented in the compositional ASRL as movement commands. However, this paradigm can be abstracted away from the language learning; i.e. the slots can be thought of as representing categories other than action related ones. The simulated robot is simply learning to set parameters for a call to some output routine. While we think of the slots as representing action words, speed words and duration words (in that order), the slots could just as easily represent other categorizations.

## 6.3 Results and discussion

| | Command given to robot | | | Robot's action | | |
|---|---|---|---|---|---|---|
| 1. | left | slow | long | left | slow | short |
| 2. | left | slow | long | left | fast | long |
| 3. | left | slow | long | right | slow | long |
| 3. | left | slow | long | straight | fast | short |
| 4. | left | slow | long | left | fast | short |
| 5. | left | slow | long | straight | slow | long |
| 6. | left | slow | long | right | fast | long |
| 7. | left | slow | long | left | slow | short |
| 8. | left | slow | long | left | fast | long |
| 9. | left | slow | long | left | slow | short |
| 10. | left | slow | long | straight | fast | long |
| 11. | left | slow | long | right | slow | short |
| 12. | left | slow | long | straight | slow | short |
| 13. | left | slow | long | right | fast | long |
| 14. | left | slow | long | left | slow | long |
| 15. | left | slow | long | right | slow | long |
| 16. | left | slow | long | straight | slow | long |
| 17. | left | slow | long | left | slow | long |
| 18. | left | slow | long | left | slow | long |
| 19. | right | fast | short | left | fast | short |
| 20. | right | fast | short | straight | slow | long |
| 21. | right | fast | short | right | slow | short |
| 22. | right | fast | short | left | fast | long |
| 23. | right | fast | short | straight | slow | long |
| 24. | right | fast | short | right | slow | short |
| 25. | right | fast | short | left | fast | long |
| 26. | right | fast | short | straight | slow | long |
| 27. | right | fast | short | right | fast | short |
| 28. | right | fast | short | right | fast | short |
| 29. | left | fast | short | left | fast | short |
| 30. | right | slow | long | right | slow | long |
| 31. | left | slow | short | left | slow | short |
| 32. | right | fast | long | right | fast | long |
| 33. | straight | fast | long | right | fast | long |
| 34. | straight | fast | long | straight | fast | long |
| 35. | straight | fast | long | left | fast | long |
| 36. | straight | fast | long | straight | fast | long |

Table 6.1: Sample run of compositional system using phrase reinforcement. After the robot learns all of the existing words in the language, a new word is added. Notice that once the word is learned, the robot can use this word with any of the other previously learned words to form new concepts.

Table 6.1 shows a sample learning session in the compositional system using phrase reinforcement. In steps 1–18, the robot is learning "left slow long". The robot then learns "right fast short" in steps 19–28. If we had learned these two concepts in the basic ASRL, then gave the robot the instruction "left fast short", the robot would need to start from scratch to learn this concept. However, in the compositional experiments, learning "left fast short" is trivial in this case. The robot learned the meaning of "left" when it learned the concept "left slow long". It also learned the meanings of "fast" and "short" when it learned "right fast short". Now the robot can use this knowledge to immediately act correctly when it sees "left fast short" for the first time. Steps 29–32 show that the robot is able to correctly respond to novel concepts that are built using previously learned words. In steps 33–36, we see the robot needing to learn a new word "straight" in the concept "straight fast long". Since the robot knows the meanings of "fast" and "long", it need only learn "straight". In the basic experiment, the robot would have had to learn this concept monolithically and we would expect it to take approximately as long as it took to initially learn "left slow long" or "right fast short".

Table 6.2 demonstrates the learning of 7 words placed over 3 slots. There are three words that can go in the first slot representing the direction to move in; these are "straight", "left" and "right". The second slot representing speed can contain "fast" or "slow". Finally, the last spot representing duration can contain "short" or "long". Additionally, any slot may be left blank as long as all three slots are not left blank. There are $[(3+1)(2+1)(2+1)]-1 = 35$ possible concepts that this language can represent using seven words in the distribution described above.

In steps 1–4, the robot only learns the meaning of "left" since the robot gets the command "left (blank) (blank)". While the robot gets positive reinforcement in step 2 for correctly executing left, it continues to explore the space by trying straight in step 3. Once it has fully explored all possible outputs, it will continue to select left since that results in the best reinforcement value. Similarly, the robot learns "right" in steps 5–8. Then in step 9, the robot gets the command "left slow (blank)". Since it already knows the meaning of left, it needs only to learn slow now.

| | Command given to robot | | | Robot's action | | |
|---|---|---|---|---|---|---|
| 1. | **left** | – | – | right | – | – |
| 2. | left | – | – | *left* | – | – |
| 3. | left | – | – | straight | – | – |
| 4. | left | – | – | **left** | – | – |
| 5. | **right** | – | – | left | – | – |
| 6. | right | – | – | *right* | – | – |
| 7. | right | – | – | straight | – | – |
| 8. | right | – | – | **right** | – | – |
| 9. | left | **slow** | – | left | *slow* | – |
| 10. | left | slow | – | left | fast | – |
| 11. | left | slow | – | left | **slow** | – |
| 12. | left | slow | **long** | left | slow | *long* |
| 13. | left | slow | long | left | slow | short |
| 14. | left | slow | long | left | slow | **long** |
| 15. | right | slow | long | right | slow | long |
| 16. | right | **fast** | long | right | *fast* | long |
| 17. | right | fast | long | right | slow | long |
| 18. | right | fast | long | right | **fast** | long |
| 19. | right | fast | **short** | right | fast | long |
| 20. | right | fast | short | right | fast | *short* |
| 21. | left | fast | short | left | fast | **short** |
| 22. | **straight** | fast | short | left | fast | short |
| 23. | straight | fast | short | *straight* | fast | short |
| 24. | straight | fast | short | right | fast | short |
| 25. | straight | fast | short | **straight** | fast | short |

Table 6.2: Sample run of compositional system using phrase reinforcement in which there are 35 concepts that can be built up from 3 direction words, 2 speed words and 2 duration words. The 7 words are learned within 25 iterations; after these 25 iterations, all 35 concepts are known. The introduction of a new word in a command is **bold**. The first time the command is properly executed is in *italics*; the action is **bold** once the robot has learned the meaning. Note that the learning algorithm we are using requires an exploration through the entire state space before the agent is convinced that it has learned the best action.

| Number of | | | | Phrase reinforcement | | | Word reinforcment | | |
|---|---|---|---|---|---|---|---|---|---|
| Action Words | Speed Words | Duration Words | Concepts | Average | Min | Max | Average | Min | Max |
| 10 | 1 | 1 | 43 | 392.23 | 283 | 574 | 400.95 | 276 | 598 |
| 8 | 2 | 2 | 80 | 270.17 | 161 | 519 | 245.94 | 160 | 364 |
| 6 | 5 | 1 | 83 | 206.61 | 134 | 316 | 130.13 | 88 | 204 |
| 6 | 3 | 3 | 111 | 188.04 | 103 | 324 | 134.74 | 91 | 211 |
| 5 | 3 | 4 | 119 | 169.13 | 114 | 278 | 93.29 | 58 | 154 |
| 4 | 4 | 4 | 124 | 160.11 | 71 | 268 | 68.47 | 45 | 96 |

Table 6.3: Each of the languages learned has 12 words. We see that the distribution of these words over the three word slots makes a great difference both learning times and number of concepts that can be represented.

When we use phrase reinforcement, blanks speed up the learning time since they allow the reinforcement to cover fewer non-blank slots. In steps 1–4, the robot was basically receiving word reinforcement for the word "left".

After 25 learning steps, the robot has learned all 7 words. It is now able to understand all of the 35 possible concepts that can be represented using the learned words in phrases.

### 6.3.1 Distribution of words over slots

Table 6.3 shows that the number of concepts that an ASRL can represent increases as the words in the language are distributed evenly among the word slots. Additionally, it shows that the learning times decrease as the number of concepts increase. The even distribution of words among slots requires the robots to search smaller, evenly distributed spaces rather than one large slot space with two small slot spaces.

The graph in figure 6-1 shows the results of learning times for a compositional ASRL and the number of concepts that can be represented in the ASRL with balanced word distribution. While learning is still exponential in the number of words in the language, empirical data seems to show that the the learning time divided by the number of concepts in a balanced distribution is a constant. The points graphed in

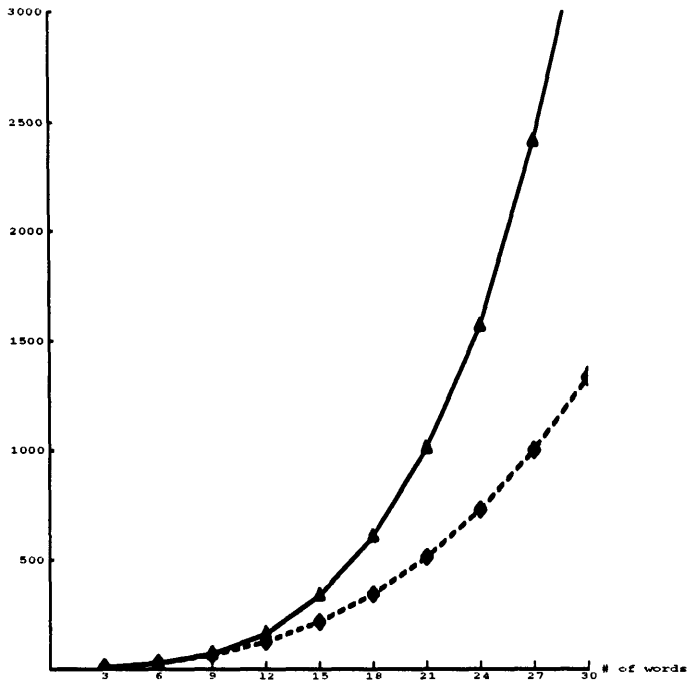



Figure 6-1: The dashed line gives the maximum number of concepts that can be represented by a language by using even distribution of the words over three word slots in the grammar. The solid line graphs average learning times for these balanced languages using phrase reinforcement.

figure 6-1 are given in table B.9.

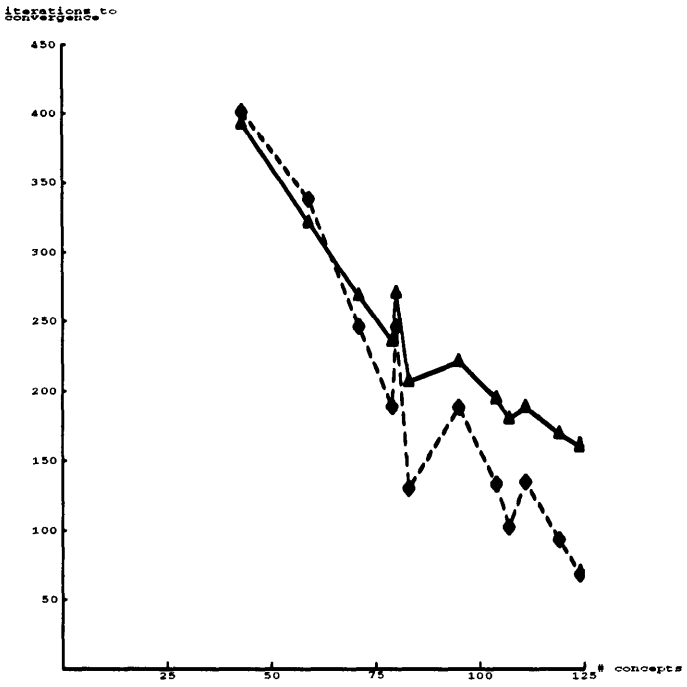
### 6.3.2 Phrase reinforcement vs. word reinforcement

As discussed above in the implementation section, phrase reinforcement is similar to task-based reinforcement and word reinforcement is similar to individual reinforcement. This learning problem is not dependent, since the meaning of a word in a word slot will only set its corresponding parameter. However, this is a result of a simplification we made rather than a characteristic of the problem. In another version of the compositional problem, two word slots might interact to set one parameter.

Since one or two of the three word slots can be left blank, phrase reinforcement gains some of the benefit of word reinforcement. For example, in the case where the robot gets the command "spin (blank) (blank)", the robot receives a reinforcement value that is only contingent upon the one word. So, we expect that phrase reinforcement and word reinforcement have learning times more closely related than in

| Concepts | Number of Action Words | Speed Words | Duration Words |
|---|---|---|---|
| 43 | 10 | 1 | 1 |
| 59 | 9 | 2 | 1 |
| 71 | 8 | 3 | 1 |
| 79 | 7 | 4 | 1 |
| 80 | 8 | 2 | 2 |
| 83 | 6 | 5 | 1 |
| 95 | 7 | 3 | 2 |
| 104 | 6 | 4 | 2 |
| 107 | 5 | 5 | 2 |
| 111 | 6 | 3 | 3 |
| 119 | 5 | 3 | 4 |
| 124 | 4 | 4 | 4 |

(a)




(b)

Figure 6-2: Learning times for 12 word compositional ASRLs. (a) Distribution of the 12 words over the three slots in the grammar that give the numbers of concepts graphed in (b). (b) Graph of learning times for compositional ASRLs. The dashed line gives results for word reinforcement and the solid line graphs results for phrase reinforcement.

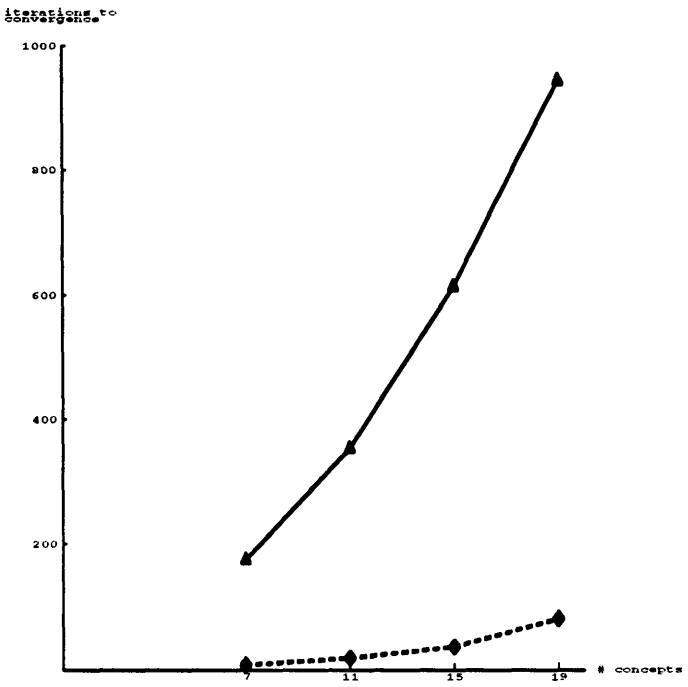



Figure 6-3: Comparison of running times for basic experiment vs. compositional experiment using task-based reinforcement.

the basic ASRL development. We see this in figure 6-2. The data points graphed in part (b) of figure 6-2 are given in table B.10. If blanks were disallowed, phrase reinforcement would take longer than it currently does, but word reinforcement would stay the same.

### 6.3.3 Compositional ASRL vs. basic ASRL

Finally, we compare the results of the compositional ASRL learning against basic ASRL learning. So that the comparisons are fair, we used the modified experimental scenario of one robot in the basic ASRL learning. The basic ASRL requires more learning time than the compositional ASRL. As we've discussed before, the basic ASRL must learn each concept as a monolithic concept. In the compositional ASRL, the robot needs only learn the words in the language to be able to understand all of the concepts. Figure 6-3 compares the two ASRL types. The data graphed in this figure is given in table B.11.

## 6.4 Future Work

The next step is to extend the work in this chapter to multiple robots. We did not have followers since our original experimental scenario would not handle the compositional ASRL well, as we discussed in the implementation section. In order for the work to be extended to multiple robots, we believe that a different experimental scenario is required.

Additionally, the work should be moved from simulation to real robots. The world is inherently compositional. There are objects in the world and ways for the objects to relate to one another, just as in a compositional language there are words and ways for words to relate to one another. This observation might provide an interested testbed for real robots.

# Chapter 7

# Related Work

## 7.1 Introduction

This work is related to several areas of artificial intelligence and computer science. The development of communication has been explored by researchers in artificial life and genetic algorithms. Multi-agent robotics is a growing research domain; most systems take the approach of providing language or trying to go without language. Some work has been done in the area of multi-agent reinforcement learning. In this chapter, we will give an overview of these areas and discuss how this work relates to previous research.

## 7.2 Artificial life: evolving communication

In the artificial life community, researchers have investigated the evolution of communication in simulated creatures. Artificial life is concerned with discovering how organisms can evolve and adapt in different environments. (For examples of work done in this field, see [Langton, 1989], [Langton *et al.*, 1991], [Langton, 1994].) Creatures are simulated by a computer program and set loose to survive in a simulated world. Most creatures select from an action set based upon an evolved mechanism for choosing actions. The creatures that evolve better selection mechanisms will be more prosperous in the simulated world. The measure of how well a creature survives in

the world is its fitness. Actions usually include movement, eating (or getting energy from some source), reproduction (usually by splitting and with mutation at the time of splitting to introduce new genes into the system) and sensing the environment. Not all systems include these capabilities, and some systems have additional capabilities such as communication.

Language development in artificial life systems tends to occur only when a new creature is being split off by one or more parents; i.e. creatures can not adapt their languages within their lifetimes. More effective languages result in creatures that have a better ability to survive in the world and have offspring. The languages are slightly mutated at the time of splitting to introduce new languages into the pool.

MacLennan describes a program of "synthetic ethology", which he defines as the creation of simulated worlds and organisms for the purpose of studying the development of behaviors [MacLennan, 1991, MacLennan and Berghardt, 1993]. One of these behaviors is communication. He describes a world where each creature has a local environment that only it can sense. The only way for a creature to predict another creature's situation is by using a broadcast message from that other creature. Creatures that can predict situations of other creatures using the communication signal have a higher fitness. There is a global environment for communication; each creature can write a symbol to the global environment, but only one symbol can appear at a time. (This writing to the global environment is similar to communication using shared memory in robots in [Arkin *et al.*, 1993].) Communicating and listening effectively adds to the fitness of a creature — and only the most fit creatures get to breed while the least fit creatures die. There is some basic adaptation of the language during the lifetime of the creature in this system; if the creature does not interpret a signal correctly, it will update its learning table to reflect the correct interpretation for the signal.

There is no adaptation within the lifetime of creatures in [Werner and Dyer, 1991] — language is genetically hard-coded. In this work, creatures must communicate in order to breed and carry on their genetic line. Since the males in the simulation are blind and the females are immobile, the males need to listen to females for directions

on how to reach them. The male needs to interpret the signal it hears; it can not sense where the sound is coming from. The relationship between the sounds a female makes and the male's actions in response to these sounds is arbitrarily based on the genomes of each creature. The creatures that can communicate effectively will breed and their language will propagate.

Artificial life systems tend to view the simulations in terms of many generations of creatures. In the robotics world, we can not afford to have multiple generations of robot hardware, although we can have multiple generations of robot programs if the robots learn while they are running. The death of millions of artificial creatures is no problem when you can restart the simulation or just type a command to create more; when using robots, the loss of just one robot can be very costly, both in terms of money and time. However, the work in communication development in artificial systems is still relevant for robotics if we look at multiple generations as multiple learning steps within a lifetime.

## 7.3 Multi-agent reinforcement learning

Whitehead [Whitehead, 1991] provides an analysis of complexity in multi-agent reinforcement learning. The issue of multiple agent reinforcement learning has also been addressed in [Tan, 1993].

Tan defines three ways that agents can communicate to cooperate using reinforcement learning. The first way is by communicating instantaneous information, such as a sensation, an action taken or a reward received. The second way is by transmitting episodes. An episode is a triple containing a sensation, an action and a reward. The final way is by communicating policies, which are learned behaviors. Sharing of policies is limited to homogeneous agents since heterogeneous agents will most likely not have the same sensors, sensor readings or action possibilities. However, heterogeneous agents can share episodes if they have the ability to interpret them. The work discussed in this thesis falls into the category of communicating instantaneous information, whether it is an instruction or a reinforcement value.

Tan shows that groups can outperform single agents but that learning can be slower initially. Sharing sensation helps if the information can be used efficiently. Sharing of learned events (episodes) and policies speeds up learning, but with the added cost of communication. If it takes longer to communicate than to learn, it would be foolish to use this method; however, it is usually much quicker to share a policy than to learn one individually from scratch. Tan states, "[I]f cooperation is done intelligently, each agent can benefit from other agents' instantaneous information, episodic experience, and learned knowledge."

## 7.4 Multi-agent robotics

Multi-agent robotics is a relatively new field, with most research done in the last five years. Only recently has there been a workshop solely on multiple robotic systems at one of the major artificial intelligence conferences [*IJCAI*, 1993]. Many different issues have been explored, but here we will concentrate on communication issues.

There are several approaches to communication in multi-agent robot systems. Some researchers believe that communication is not necessary. Other researchers rely on implicit communication. Some use communication, but provide the robots with the language to be used for communication. To our knowledge, no other systems have explored the development of languages by robots.

Figure 7-1 presents pros and cons for four different means for robots to begin communicating. The work in this thesis has concentrated on the third option. Most work in the field concentrates on the first option.

An example of a provided communication language is [Parker, 1994, Parker, 1993]. This work investigates the use of heterogeneous multiple agents to complete tasks more effectively than a single super-robot. This system uses communication in the form of broadcast messages. A robot about to attempt a task will broadcast that it is about to do the task. Other robots that hear this message can use the information to decide that they should go off and try a different task than the one that the other robot is attempting to do. The communication language is completely built in;

1. Provide the robots with a language

    - Fastest startup on robots (ignoring how long it takes a person to program it)
    - May not be well-suited to robots or tasks
    - Not adaptable by robots
    - High human involvement both in the initial development of the language and in making any necessary adaptations to the language according to needs of the robots

2. Bootstrap with a human language and give robots ability to adapt

    - Fast startup, but adaptation may require a good deal of time if the programmer did not anticipate the needs of the robots well
    - Can adapt to suit robots and tasks
    - High human involvement initially, but less involvement once the robots are adapting the language themselves

3. Allow the robots to develop their language

    - Slower starting time (when robots start, they need to start learning a language rather than having an initial human provided language to start up with)
    - Result will be well-suited to robots and tasks
    - Little to no human involvement

4. Bootstrap with a robot developed language and give robots ability to adapt

    - Fast startup
    - Since language was developed by other robots, probably will not require much adaptation at the outset
    - Adaptable
    - Much less human involvement

Figure 7-1: This table compares four different methods for providing robots with a basis for communication.

however, the language is not required for the task to be completed. [Parker, 1994] shows that even with the breakdown of communication, the tasks can be completed, but not as efficiently. For other examples of systems that use provided languages, see [Fukuda and Kawauchi, 1990], [Matsumoto *et al.*, 1990], [Shin and Epstein, 1985], [Arkin *et al.*, 1993], [Arkin and Hobbs, 1993], and [Weiß, 1993]

Other systems use no communication between the robots. For example, Matarić [Matarić, 1993a, Matarić, 1994, Matarić, 1993b] discusses the building up of large social behaviors through the use of low level primitives. There is no communication in this system. The robots broadcast their locations, but there is no discussion of what any robot should do next.

Communication between agents can either be implicit or explicit. In implicit communication, an agent needs to deduce what is being communicated without the benefit of being told directly. [Huber and Durfee, 1993] uses implicit communication in his plan recognition system. One robot observes another robot moving through the world and tries to determine what goal the other robot is trying to reach. Some have argued that implicit communication is preferable to explicit communication since the environment may be hostile to communication (e.g. in a war, tanks would not want to broadcast movements). However, plan recognition is very difficult and robots can fail to recognize what the other robots are doing. While there are still bound to be some misunderstandings in explicit communication, there is a greater chance for the message to get across to the other robots. Clearly, we fall into the camp that advocates explicit communication.

## 7.5 Distributed Artificial Intelligence

The Distributed Artificial Intelligence community has explored many of the same issues that are now currently being studied in the area of multiple robotics. Some survey articles are [Bond and Gasser, 1988], [Durfee *et al.*, 1992], [Decker, 1987], [Durfee *et al.*, 1989] and [Decker, 1987].

In DAI, problems are solved by breaking them down into parts to be solved by

multiple agents, usually software agents. Cooperation between agents is necessary since it is difficult to break down problems into independent portions. DAI allows for systems to adapt to failures of components. Communication can take the form of message passing between agents. Another means for communication is blackboarding; agents post messages to a central area for other agents to read. Our language is more like blackboarding. The leader does not send signals to specific robots; instead, it broadcasts the signal for all of the followers to hear.

# Chapter 8

# Conclusion

In this thesis we demonstrated that robots can learn languages for communication. These languages that are developed by the robots are adaptable and allow the robots to continue communicating in the face of changing environments.

We showed that task-based reinforcement is an effective reinforcement technique for dependent tasks and discussed why the individual reinforcement method failed for our tasks and how it can fail in other multi-agent learning tasks.

The basic ASRL development demonstrated that learning times are exponential in the number of robots and in the number of concepts. We also showed that giving robots information about the actions of other robots does not help the learning problem; in fact, it slows down the learning.

The development of context dependent ASRLs showed that robots can use information about the world state to determine the meaning of signals in the ASRL. Since fewer signals are required to represent concepts, the context dependent language can be developed in less time than the basic ASRL.

The learning of compositional ASRLs demonstrated that adding a small grammar into the language learning problem allows us to have a large number of concepts by learning a relatively small number of words to build into phrases.

# Appendix A

# Robots

## A.1 Introduction

The ASRL development experiments in this thesis have been performed using both real-world robot teams and in simulation. This chapter describes the robot hardware used in this work and discusses the assumptions we have made due to limitations of the hardware.

## A.2 Hardware

Bert and Ernie, two of the robots used in this research, are Sensor Robots designed by Fred Martin at the Media Laboratory at the Massachusetts Institute of Technology [Martin and Sargent, 1991]. We have six of the Sensor Robots, but have used only three at a time in this research due to hardware failures.

Each robot is approximately $9''l \times 6''w \times 4''h$, with a single circuit board containing most of the computational and sensory resources of the robot. A 6v battery strapped to the underside of the chassis supplies the power for the robot. The robots are shown in figure A-1.

The primary computational resource is an on-board Motorola 6811 microprocessor. The programming environment is IC, a multi-tasking interactive C compiler and interpreter developed by Randy Sargent [Sargent and Martin, 1991]. IC allows a Sen-

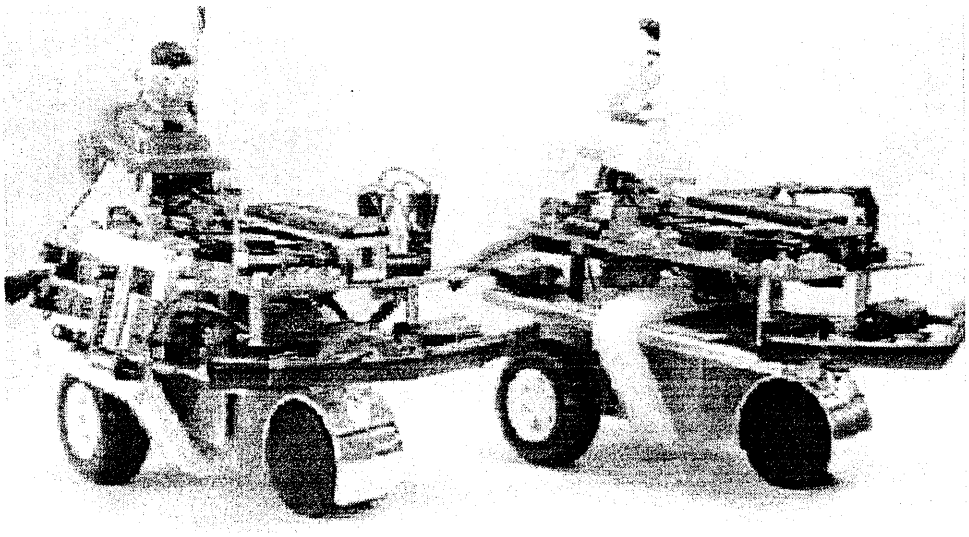


Figure A-1: Two of the Sensor Robots used: Bert and Ernie

sor Robot to be addressed through a serial line from a host computer as well as the downloading of programs for autonomous activity. The work described in this thesis was implemented with the robots under autonomous control.

Locomotion is controlled by a dual geared-wheel drive stripped from a Radio Shack Red Fox Racer. The direction of the robot is controlled by varying the speeds of the left and right motors (with negative speed moving the motor backwards). The two motorized wheels are at the rear of the robot chassis and a caster is on the front.

Communication from human to the robots is through an infra-red remote control transmitter. The robots use infra-red receivers similar to those found in televisions and VCRs. Since one robot is designated as a leader and is the only robot that should hear the human signal, we prevent the other robots from using data from their infra-red receivers.

The robots communicate between themselves using a pair of radio transmitter and receiver boards similar to those used in garage door openers. (The transmitter and receiver each run off of a separate 9v battery.) The radio boards have a communication range of about 15 feet.

In addition to the infra-red and radio receivers, the sensor robots contain four (front and rear, left and right) bump sensors, left and right shaft encoders, an inclination sensor, photosensitive cells, a microphone, and infra-red emitters. These

additional sensory abilities of the robots were not substantively used in the experiments described here. Additionally, each robot has a speaker and a 16-character LCD, both used primarily for debugging and monitoring of the robot's activity.

## A.3 Programmed robot abilities

We have programmed in many different action routines for the robots to utilize while learning to communicate. These actions fall into three primary categories: action-based tasks, communication-based tasks and reinforcement-based tasks.

Action-based tasks are procedures that tell the robot how to control its motors to move in certain directions, such as "move straight", "spin", "turn right", "turn left" and "back up." Each action command is performed for a set amount of time.

Communication-based tasks include "send signal", "wait for signal" and "determine meaning of signal." Reinforcement-based tasks include "listen for reinforcement", "update reinforcement tables" and "select action." (Reinforcement learning was discussed in Chapter 3.)

## A.4 Impact of hardware limitations on research

The robots have no vision capability, which severely restricts their ability to recognize the actions of other robots. Since the robots are unable to recognize the actions that other robots take, reinforcement is provided by a human instructor. In lieu of human reinforcement, the robots could broadcast the action performed after completion; however, in designing these experiments, it was decided that there would be too many messages that would raise issues of turn-taking.

We decided to have the robots communicate using the radio boards because they were available. However, robots can communicate in many other ways. Any incoming sensor data can be taken as a communication. For example, a team of office-cleaning robots can communicate that a task is done simply by performing the task effectively. If one robot empties the trash in an office, another office can tell that the task is

completely by sensing an empty trash can. The robot that emptied the trash could have also sent a signal to say "I emptied the trash." Due to the limited sensing capabilities of the Sensor Robots, the radio boards are a more effective means of communication.

Since the robots are blind, they also can not recognize that another robot is another robot. Instead of requiring robots to blast infra-red signals to make the other robots aware of their presence, we assume that radio signals can only be broadcast by another robot; i.e. the robots recognize their kin by the signals they send.

In order to continue to operate with the above assumptions, we have built in finite state control into the robots. Each robot will sit until it hears a signal which it needs to act upon. After acting, the robots will sit and wait to receive a reinforcement signal.

# Appendix B

# Data tables and graphs

This appendix contains data tables for points graphed in Chapters 4, 5 and 6.

| Size of Language | Number of Iterations to Convergence | | |
|---|---|---|---|
| | Average | Minimum | Maximum |
| 2 | 15.34 | 10 | 24 |
| 3 | 110.30 | 33 | 501 |
| 4 | 340.38 | 53 | 990 |
| 5 | 906.62 | 255 | 2472 |
| 10 | 15011.61 | 2868 | 51031 |
| 20 | 232267.82 | 44196 | 1241767 |

Table B.1: Learning times for a two member troupe using task-based reinforcement. Experiments for each language size were run 100 times. These points are graphed in figures 4-4 and 4-2.

| Size of Language | Number of Iterations to Convergence Average | Minimum | Maximum |
|---|---|---|---|
| 2 | 27.21 | 10 | 80 |
| 3 | 327.71 | 35 | 1211 |
| 4 | 1530.12 | 340 | 6666 |
| 5 | 4415.60 | 652 | 17533 |
| 10 | 163530.62 | 37130 | 705029 |
| 20 | 5105434.96 | 100000 | 18544822 |

Table B.2: Data above is for a three member troupe and was collected over 100 runs for each language size using task-based reinforcement. These points are graphed in figures 4-2 and 4-3.

| Size of Language | Number of Iterations to Convergence Average | Minimum | Maximum | Number Failing to Converge |
|---|---|---|---|---|
| 2 | 17.82 | 12 | 31 | 0 |
| 3 | 52.16 | 24 | 100 | 0 |
| 4 | 125.66 | 55 | 304 | 0 |
| 5 | 238.97 | 103 | 580 | 0 |
| 10 | 2086.83 | 780 | 5972 | 0 |
| 20 | 19584.40 | 8150 | 57915 | 0 |

Table B.3: Two robots being separately reinforced. These points are graphed in figure 4-1.

| Size of Language | Number of Iterations to Convergence Average | Minimum | Maximum | Number Failing to Converge |
|---|---|---|---|---|
| 2 | 23.51 | 13 | 43 | 0 |
| 3 | 94.1 | 33 | 794 | 2 |
| 4 | 267.69 | 102 | 2507 | 4 |
| 5 | 542.77 | 192 | 3484 | 6 |
| 10 | 7454.17 | 2997 | 24309 | 71 |
| 20 | 164117.15 | 32297 | 762272 | 445 |

Table B.4: Three robots being separately reinforced, cut at 1,000,000. These points are graphed in figure 4-3.

| Size of Language | Number of Iterations to Convergence | | |
|---|---|---|---|
| | Average | Minimum | Maximum |
| 2 | 521.14 | 189 | 1462 |
| 3 | 2675.72 | 919 | 7151 |
| 4 | 6906.20 | 2496 | 14775 |
| 5 | 14270.41 | 5510 | 31077 |
| 10 | 101254.71 | 51663 | 194834 |
| 20 | 509240.32 | 100000 | 1179046 |

Table B.5: Two robots with knowledge of what the other robot has done. These points are graphed in figure 4-4.

| Number of Robots | Number of Iterations to Convergence | | |
|---|---|---|---|
| | Average | Minimum | Maximum |
| 2 | 118.83 | 28 | 356 |
| 3 | 326.31 | 71 | 1264 |
| 4 | 893.43 | 109 | 2445 |
| 5 | 2891.32 | 131 | 12350 |
| 6 | 9488.12 | 817 | 39365 |
| 7 | 29207.23 | 1572 | 100209 |
| 8 | 73019.25 | 5622 | 263424 |

Table B.6: Results of varying troupe size from 2 to 10 robots for the development of a three element basic ASRL using task-based reinforcement. These points are graphed in figure 4-5.

| Number of Robots | Number of Iterations to Convergence | | |
|---|---|---|---|
| | Average | Minimum | Maximum |
| 2 | 374.20 | 67 | 1424 |
| 3 | 1362.87 | 229 | 4416 |
| 4 | 6565.58 | 992 | 26813 |
| 5 | 20659.19 | 2199 | 75002 |
| 6 | 90182.94 | 10014 | 439304 |
| 7 | 390228.21 | 26914 | 1599106 |
| 8 | 1565770.83 | 218865 | 5728755 |

Table B.7: Results of varying troupe size from 2 to 10 robots for the development of a four element language. These points are graphed in figure 4-5.

| Number of | | Number of Iterations to Convergence | | |
|---|---|---|---|---|
| Signals | Sensor Values | Average | Minimum | Maximum |
| 12 | 1 | 31672.23 | 6719 | 133106 |
| 6 | 2 | 21929.58 | 6610 | 74519 |
| 4 | 3 | 15146.98 | 3456 | 32487 |
| 3 | 4 | 11850.23 | 3969 | 24809 |
| 2 | 6 | 9482.71 | 4033 | 25759 |
| 1 | 12 | 5907.16 | 2516 | 13875 |

Table B.8: Results for the development of 12 concept context dependent ASRLs using varying numbers of sensor values. These points are graphed in figure 5-1.

| Number of | | | | Phrase reinforcement | | |
|---|---|---|---|---|---|---|
| Action Words | Speed Words | Duration Words | Concepts | Average | Min | Max |
| 1 | 1 | 1 | 7 | 8.27 | 4 | 15 |
| 2 | 2 | 2 | 26 | 27.19 | 14 | 47 |
| 3 | 3 | 3 | 63 | 69.12 | 40 | 130 |
| 4 | 4 | 4 | 124 | 160.11 | 71 | 268 |
| 5 | 5 | 5 | 215 | 336.35 | 207 | 562 |
| 6 | 6 | 6 | 342 | 605.03 | 378 | 1047 |
| 7 | 7 | 7 | 511 | 1006.82 | 608 | 1573 |
| 8 | 8 | 8 | 728 | 1563.19 | 966 | 2268 |
| 9 | 9 | 9 | 999 | 2408.83 | 1523 | 3487 |
| 10 | 10 | 10 | 1330 | 3461.95 | 2073 | 5231 |

Table B.9: Learning times for compositional ASRLs with balanced word distribution using phrase reinforcement. These points are graphed in figure 6-1.

| Number of | | | | Phrase | | | Word | | |
|---|---|---|---|---|---|---|---|---|---|
| Action Words | Speed Words | Duration Words | Concepts | reinforcement | | | reinforcment | | |
| | | | | Average | Min | Max | Average | Min | Max |
| 10 | 1 | 1 | 43 | 392.23 | 283 | 574 | 400.95 | 276 | 598 |
| 9 | 2 | 1 | 59 | 321.05 | 218 | 508 | 337.84 | 219 | 537 |
| 8 | 3 | 1 | 71 | 268.22 | 182 | 412 | 246.06 | 174 | 422 |
| 7 | 4 | 1 | 79 | 235.73 | 144 | 403 | 188.87 | 122 | 331 |
| 8 | 2 | 2 | 80 | 270.17 | 161 | 519 | 245.94 | 160 | 364 |
| 6 | 5 | 1 | 83 | 206.61 | 134 | 316 | 130.13 | 88 | 204 |
| 7 | 3 | 2 | 95 | 220.75 | 155 | 363 | 188.04 | 118 | 325 |
| 6 | 4 | 2 | 104 | 194.30 | 113 | 295 | 133.20 | 79 | 257 |
| 5 | 5 | 2 | 107 | 179.69 | 95 | 263 | 102.35 | 67 | 152 |
| 6 | 3 | 3 | 111 | 188.04 | 103 | 324 | 134.74 | 91 | 211 |
| 5 | 3 | 4 | 119 | 169.13 | 114 | 278 | 93.29 | 58 | 154 |
| 4 | 4 | 4 | 124 | 160.11 | 71 | 268 | 68.47 | 45 | 96 |

Table B.10: Learning times for 12 word compositional languages with varying word distributions for both phrase and word reinforcement. These points are graphed in figure 6-2.

| Number of | Compositional ASRL | | | Basic ASRL | | |
|---|---|---|---|---|---|---|
| Concepts | Average | Min | Max | Average | Min | Max |
| 7 | 8.37 | 4 | 17 | 175.29 | 119 | 329 |
| 11 | 19.34 | 11 | 40 | 353.28 | 193 | 664 |
| 15 | 37.4 | 23 | 61 | 613.98 | 356 | 1773 |
| 19 | 82.13 | 39 | 1078 | 943.36 | 453 | 1829 |

Table B.11: Learning times for compositional ASRLs and basic ASRLs using phrase reinforcement. These points are graphed in figure 6-3.

# Bibliography

[*AAAI*, 1991] *Proceedings of the Ninth National Conference on Artificial Intelligence*, Anaheim, California, July 1991.

[Arkin and Hobbs, 1993] Ronald C. Arkin and J. David Hobbs. Dimensions of communication and social organization in multi-agent robotic systems. In Meyer et al. [Meyer *et al.*, 1993], pages 486–493.

[Arkin *et al.*, 1993] Ronald C. Arkin, Tucker Balch, and Elizabeth Nitz. Communication of behavioral state in multi-agent retrieval tasks. In *Proceedings of the 1993 International Conference on Robotics and Automation*, pages 588–594, 1993.

[Bond and Gasser, 1988] Alan H. Bond and Les Gasser, editors. *An analysis of problems and research in DAI*, chapter 1, pages 3–35. Morgan Kaufmann Publishers, Inc., 1988.

[Chapman and Kaelbling, 1991] David Chapman and Leslie Pack Kaelbling. Input generalization in delayed reinforcement learning: an algorithm and performance comparisons. In *Proceedings of the 12th International Joint Conference on Artificial Intelligence*, pages 726–731, 1991.

[Chomsky, 1986] Noam Chomsky. *Knowledge of language: its nature, origin, and use.* Praeger, 1986.

[Decker, 1987] Keith S. Decker. Distributed problem-solving techniques: a survey. *IEEE Transactions on Systems, Man and Cybernetics*, SMC-17(5):729–740, September/October 1987.

[Drescher, 1991] Gary L. Drescher. *Made-up Minds.* The MIT Press, 1991.

[Drescher, 1993] Gary L. Drescher. The schema mechanism. In S.J. Hanson, W. Remmele, and R.L. Rivest, editors, *Machine Learning: From Theory to Applications. (Cooperative research at Siemens and MIT).* Springer-Verlag, 1993. Lecture Notes in Computer Science.

[Durfee *et al.*, 1989] Edmund H. Durfee, Victor R. Lesser, and Daniel D. Corkill. Cooperative distributed problem solving. In Avron Barr, Paul R. Cohen, and Edward A. Feigenbaum, editors, *The Handbook of Artificial Intelligence*, volume VI, chapter 17, pages 83–147. Addison-Wesley, 1989.

[Durfee *et al.*, 1992] Edmund H. Durfee, Victor R. Lesser, and Daniel D. Corkill. Distributed problem solving. In Stuart C. Shapiro, editor, *Enclyclopedia of AI*, pages 379–388. John Wiley and Sons, Inc., second edition, 1992.

[Fukuda and Kawauchi, 1990] T. Fukuda and Y. Kawauchi. Communication and distributed intelligence for cellular robotic system CEBOT. In *1990 Japan-USA Symposium on Flexible Automation*, pages 1085–1092, July 1990.

[Huber and Durfee, 1993] Marcus J. Huber and Edmund H. Durfee. Observational uncertainty in plan recognition among interacting robots. In *IJCAI* [*IJCAI*, 1993], pages 68–75.

[*IJCAI*, 1993] *International Joint Conference on Artificial Intelligence Workshop on Dynamically Interacting Robots*, Chambery, France, August 1993.

[Kaelbling, 1993] Leslie Pack Kaelbling. *Learning in Embedded Systems.* The MIT Press: A Bradford Book, 1993. Revised version of Ph.D. Thesis, Stanford University, 1990.

[Langton *et al.*, 1991] C.G. Langton, C. Taylor, J.D. Farmer, and S. Rasmussen, editors. *Artificial Life II*, volume X of *Santa Fe Institute Studies in the Sciences of*

*Complexity.* Addison-Wesley, 1991. Proceedings of the 2nd interdisciplinary workshop on synthesis and simulation of living systems held in February 1990 in Los Alamos, NM.

[Langton, 1989] Christopher G. Langton, editor. *Artificial Life*, volume VI of *Santa Fe Institute Studies in the Sciences of Complexity.* Addison-Wesley, 1989. Proceedings of the interdisciplinary workshop on the synthesis and simulatino of living systems held in September 1987 in Los Alamos, NM.

[Langton, 1994] Christopher G. Langton, editor. *Artificial Life III.* Addison-Wesley, 1994.

[MacLennan and Berghardt, 1993] Bruce J. MacLennan and Gordon M. Berghardt. Synthetic ethology and the evolution of cooperative communication. *Adaptive Behavior*, 2(2):161–187, 1993.

[MacLennan, 1991] Bruce MacLennan. *Synthetic ethology: an approach to the study of communication*, pages 631–658. Volume X of Langton et al. [Langton *et al.*, 1991], 1991. Proceedings of the 2nd interdisciplinary workshop on synthesis and simulation of living systems held in February 1990 in Los Alamos, NM.

[Mahadevan and Connell, 1991] Sridhar Mahadevan and Jonathan Connell. Automatic programming of behavior-based robots using reinforcement learning. In *AAAI* [*AAAI*, 1991], pages 768–773.

[Martin and Sargent, 1991] Fred Martin and Randy Sargent. The MIT sensor robot: User's guide and technical reference. October 1991.

[Matarić, 1991] Maja J. Matarić. A comparative analysis of reinforcement learning methods. Memo 1322, Massachusetts Institute of Technology Artificial Intelligence Laboratory, Cambridge, Massachusetts, October 1991.

[Matarić, 1993a] Maja J. Matarić. Designing emergent behaviors: local interactions to collective intelligence. In Meyer et al. [Meyer *et al.*, 1993], pages 432–441.

[Matarić, 1993b] Maja J. Matarić. Kin recognition, similarity, and group behavior. In *Proceedings of the Fifteenth Annual Conference of the Cognitive Science Society*, pages 705–710, Boulder, Colorado, 1993.

[Matarić, 1994] Maja J. Matarić. *Interaction and Intelligent Behavior.* PhD thesis, Massachusetts Institute of Technology, May 1994.

[Matsumoto *et al.*, 1990] A. Matsumoto, H. Asama, and Y. Ishida. Communication in the autonomous and decentralized robot system ACTRESS. In *Proceedings of the IEEE International Workshop on Intelligent Robots and Systems*, pages 835–840, Tsuchura, Japan, July 1990.

[Meyer *et al.*, 1993] Jean-Arcady Meyer, Herbert L. Roitblat, and Stweart W. Wilson, editors. *From Animals to Animats 2:Proceedings of the Second International Conference on Simulation of Adaptive Behavior*, Honolulu, Hawaii, December 1993.

[*ML*, 1993] *Machine Learning: Proceedings of the Tenth International Conference*, Amherst, Massachusetts, June 1993.

[Mullender, 1993] Sape Mullender, editor. *Distributed Systems.* Addison-Wesley: ACM Press, 1993.

[Parker, 1993] Lynne E. Parker. Adaptive action selection for cooperative agent teams. In Meyer et al. [Meyer *et al.*, 1993], pages 442–450.

[Parker, 1994] Lynne E. Parker. *Heterogeneous multi-robot cooperation.* PhD thesis, Massachusetts Institute of Technology, February 1994.

[Pierce and Kuipers, 1991] David Pierce and Benjamin Kuipers. Learning hill-climbing functions as a strategy for generating behaviors in a mobile robot. In Jean-Arcady Meyer and Stewart W. Wilson, editors, *From Animals to Animats:Proceedings of the First International Conference on Simulation of Adaptive Behavior*, pages 327–336, Paris, France, September 1991.

[Pierce, 1991] David Pierce. Learning turn and travel actions with an uninterpreted sensiomotor apparatus. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 391–404. Lawrence Erlbaum Associates, 1991.

[Sargent and Martin, 1991] Randy Sargent and Fred Martin. IC: Multi-tasking interactive C for the 6811. IC Version 2.5, October 1991.

[Shavlik and Dietterich, 1990] Jude W. Shavlik and Thomas G. Dietterich, editors. *Readings in Machine Learning.* Morgan Kaufmann, 1990.

[Shewchuk, 1991] John P. Shewchuk. Ph.D. thesis proposal. Department of Computer Science, Brown University, Providence, Rhode Island, 1991.

[Shin and Epstein, 1985] K. Shin and M. Epstein. Communication primitives for a distributed multi-robot system. In *Proceedings of the IEEE Robotics and Automation Conference*, pages 910–917, 1985.

[Sutton, 1984] Richard S. Sutton. *Temporal credit assignment in reinforcement learning.* PhD thesis, University of Massachusetts, February 1984.

[Sutton, 1988] Richard S. Sutton. Learning to predict by the methods of temporal differences. *Machine Learning*, 3:9–44, 1988.

[Sutton, 1992] Richard S. Sutton, editor. *Machine Learning: Special issue on reinforcement learning*, volume 8:3–4. May 1992.

[Tan, 1993] Ming Tan. Multi-agent reinforcement learning: independent vs. cooperative agents. In *ML* [*ML*, 1993], pages 330–337.

[Tesauro, 1992] Gerald Tesauro. Practical issues in temporal difference learning. In Sutton [Sutton, 1992], pages 33–53.

[Watkins and Dayan, 1992] Christopher J.C.H. Watkins and Peter Dayan. Q-learning (technical note). In Sutton [Sutton, 1992], pages 55–68.

[Watkins, 1989] Christopher J.C.H. Watkins. *Learning from Delayed Rewards.* PhD thesis, King's College, May 1989.

[Weiß, 1993] Gerhard Weiß. Action selections and learning in multi-agent environments. In Meyer et al. [Meyer *et al.*, 1993], pages 502–510.

[Werner and Dyer, 1991] Gregory M. Werner and Michael G. Dyer. *Evolution of communication in artificial organisms*, pages 659–687. Volume X of Langton et al. [Langton *et al.*, 1991], 1991. Proceedings of the 2nd interdisciplinary workshop on synthesis and simulation of living systems held in February 1990 in Los Alamos, NM.

[Whitehead, 1991] Steven D. Whitehead. A complexity analysis of cooperative mechanisms in reinforcement learning. In *AAAI* [*AAAI*, 1991], pages 607–613.

[Yanco and Stein, 1993] Holly Yanco and Lynn Andrea Stein. An adaptable communication protocol for cooperating mobile robots. In Meyer et al. [Meyer *et al.*, 1993], pages 478–485.